Models and optimization approaches for flexible job shop scheduling with lot sizing

Andrzej Bożek
Faculty of Electrical and Computer Engineering, Rzeszow University of Technology,
al. Powstańców Warszawy 12, 35-959 Rzeszów, Poland
email: abozek@kia.prz.edu.pl

Frank Werner
Faculty of Mathematics, Otto-von-Guericke-University,
PSF 4120, 39016 Magdeburg, Germany
email: frank.werner@ovgu.de

December 4, 2016

Abstract: Models and optimization approaches are developed for a flexible job shop scheduling problem with lot streaming and lot sizing of the variable sublots. A two-stage optimization procedure is proposed. First, the makespan value is minimized with the smallest sublots defined for the problem instance. This makes it possible to shorten the makespan significantly, because each sublot is transferred separately to the next operation of a job. In the second stage, the sizes of the sublots are maximized without increasing the obtained makespan value. In this way, the number of sublots and transport activities is limited together with the related manufacturing cost. Two objectives are defined for the second stage. The first one is the maximization of the sum of the sublot sizes of all operations, the second one is the maximization of the number of the operations which do no need to be split at all. A mixed-integer linear programming, constraint programming, and graph-based models are implemented for the problem. Two optimization approaches are developed and compared in computational experiments for each stage and objective, one approach is based on a third-party solver, and the second one on an independent own implementation, namely a tabu search and a greedy constructive heuristic.

Keywords: flexible job shop, lot streaming, lot sizing, mixed-integer linear programming, tabu search, constraint programming

MSC: 90B35, 90C11, 90C59

1 Introduction

The flexible job shop (FJS) is one of the most popular scheduling problems. It is mainly because of the comprehensive character of its formulation, so that it relates to many real-world production environments. To increase the practical usability of the JSP even more, it is often extended by additional features, e.g., setup and transport times. If scheduling without imposed due dates is considered, the makespan is the typically used objective function. It is not only a conceptually clear function, but it has also the property of maximizing machine utilization by reducing idle intervals in a schedule. However, the level of the optimization of machine utilization is inherently limited in classic non-preemptive JSP environments. Real-world manufacturing systems are more and more often technically and organizationally prepared to employ advanced techniques of improving the utilization of resources, for example, preemptive operation changes, partial overlapping of consecutive processes, lot streaming with transfer batches. These techniques require extended scheduling methods.

In Figure 1, a collation of the scheduling and lot sizing approaches considered in this work is presented. It is based on the best results obtained for the benchmark instance 4, used for the computational experiments (Subsection 7.1), and chosen here as an example (for the detailed explanation of the symbols used, see the following description). Each rectangle on the Gantt charts represents a sublot, i.e., a smallest indivisible and uninterruptible processing unit. The sublots belonging to a single operation are marked by the same gray scale level. The rectangles drawn with bold borders indicate the sublots constituting the critical path of the schedule. The schedule represented by Gantt chart 1 is a solution of the classic flexible job shop scheduling problem without lot streaming. It consists of $N = 34$ operations which can also be considered as sublots of maximal sizes. The sum of the sizes of these sublots is $S = 3602$. In Gantt chart 2, the solution of the same instance is shown, where in addition lot streaming has been introduced. Each operation is divided into 8 to 24 sublots. The size and number of sublots is, in general, different for consecutive operations of the same job, thus, it is the case of *variable sublots*. However, the sizes of the sublots of a single operation are equal, except for possibly the last one, which complements the total size of the related job. This scenario embodies a typical manufacturing organization, where lot streaming and transfer batching are applied to improve some production indicator – the makespan in the considered case. In general, the smaller the sublot sizes, the makespan can be potentially better minimized, but some lower bounds on these sizes have to exist, determined by technological and organizational constraints, e.g., minimal batch sizes accepted by a particular process, properties of transport devices, and so on. The schedule 2 (Fig. 1) is assumed to be obtained with such smallest permitted sublot sizes. The sum of the sublot sizes equals 251 and the total number of sublots is equal to 532. As the result of lot streaming, the makespan value has decreased by 19%. Lot streaming is applied in order to minimize the makespan value, but it contributes its own cost, e.g., related to a growing number of transport operations, so the sublot sizes should be not smaller than necessary. This is the motivation to introduce the next optimization stage in which the sum of the sublot sizes is maximized without increasing the makespan value. In this stage, the order of processing is not changed, and only lot sizing is performed. The result is presented in Gantt chart 3. It was possible to increase the sum of the sublot sizes from 251 to 1942 and decrease the number of sublots from 532 to 152. Additionally, 11 of the 34 operations are not split at all. The non-split operations are indicated by stars and for the other operations the number is given, instead of the star, which specifies the ratio of the maximized sublot size from the schedule 3 to the minimal sublot size from the schedule 2. The non-split operations have the important feature that the whole processing outcome can be transferred to the next process after the operation is finished. Any partial processing result (sublot) can be, of course, also transferred without increasing the makespan value. Therefore, the non-split operations introduce a significant flexibility to the schedule execution, making it possible to process such operations without lot streaming or with the use of sublots of any size. Because of that, one might be interested in the maximization of the number of such operations. The result of scheduling with the use of this objective is shown in Gantt chart 4 (Fig. 1). This objective makes it possible to obtain a number of non-split operations equal to or greater than in the case of maximizing the sum of the sublot sizes.

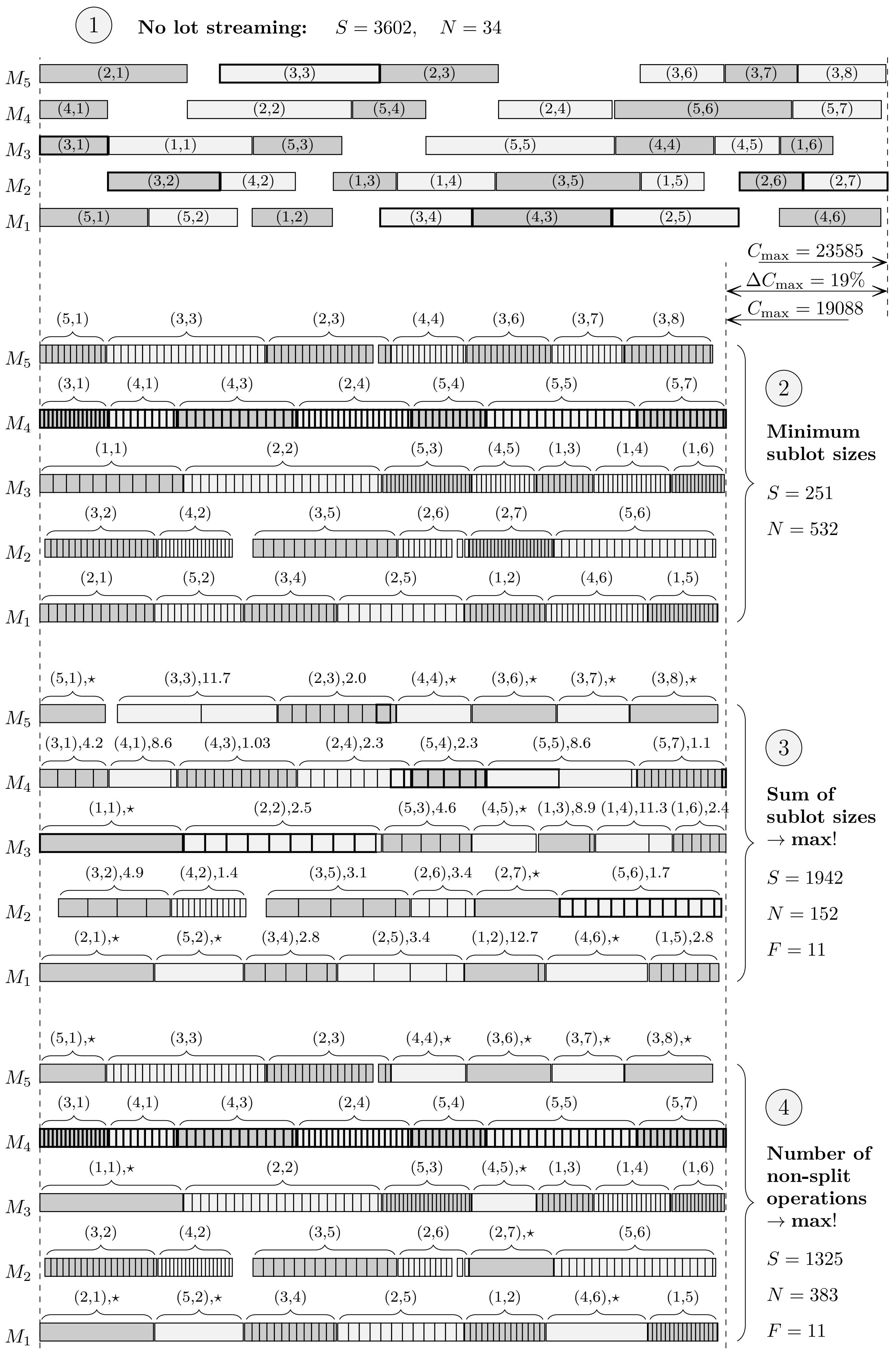


Figure 1: Comparison of the considered scheduling and lot sizing approaches

Additionally, more computationally effective algorithms can be implemented for this objective, especially when a simplified optimization procedure is used under the assumption that the sublot size can be only either minimal or maximal.

In this work, a comprehensive set of approaches is proposed which support the two-stage scheduling and lot sizing optimization described above. In the first stage, the makespan value is used as the scheduling objective. For the second stage, two variants of lot sizing objectives are considered, namely the sum of the sublot sizes and the number of non-split operations. For each objective, at least one approach using a commercial solver, as well as an individually written algorithm have been prepared and compared.

Some related work is discussed in Section 2. In Section 3, a formal problem description is provided and in Section 4, a graph model for the problem is introduced. The approaches supporting the scheduling stage are described in Section 5: a method based on constraint programming and the IBM ILOG CP Optimizer (Subsection 5.1), followed by a tabu search algorithm (Subsection 5.2). Lot sizing with maximization of the sum of the sublot sizes is discussed in Subsection 6.1: an approach based on mixed-integer linear programming (MILP) and the IBM ILOG CPLEX solver is proposed (Subsection 6.1.1), as well as a method which decomposes the problem into a linear programming task without integer variables and a top-level discrete optimization task (Subsection 6.1.2). In Subsection 6.2, lot sizing with the maximization of the number of non-split operations is considered, for which three approaches are proposed: a method based on a MILP model with continuous sublot sizes and the CPLEX solver, a similar but simplified approach where the sublot sizes can by only either minimal or maximal, and a greedy algorithm which exploits the graph model for the problem. Computational experiments are described in Section 7. They are based on a set of 20 dedicated benchmark instances (Subsection 7.1). The presentation of the settings of the experiments (Subsection 7.2) is followed by an analysis of the results (Subsection 7.3). The proposed approaches are summarized in Section 8.

2 Related work

The possibility of overlapping of consecutive operations of a job is a frequently used extension of classic scheduling problem formulations, such as the flow shop (FS) or the job shop (JS). A few variants of the implementation of this extension are used. The overlapping can be introduced by starting a successive operation of a job before the previous one is finished, provided that a required part of the previous operation is processed [1, 17]. In the case of manufacturing environments with continuous processes, the continuous flow between consecutive operations can be taken into account in scheduling [3]. Using a preemptive scheduling model is another possibility [14]. The most often used method of supporting the overlapping of operations is, however, *lot streaming* [9, 34]. When lot streaming is applied, operations (lots) are divided into smaller parts (sublots), which can be processed and transported individually. If the sublot sizes are identical for each operation of a job, the sublots are called *consistent*, otherwise they are *variable*, the consistent sublots are called *equal* if all are of the same size. Lot streaming is in the form *without intermingling*, if all sublots of each operation have to be processed one by one on a machine, or *with intermingling* otherwise.

Lot streaming is a subject of research since many years [31], but it is still studied nowadays, because of its practical importance and complexity in modeling and optimization. It is considered typically in conjunction with the flow shop [2, 8, 10, 11, 16, 25, 28, 31, 35, 36], job shop [7, 15, 24, 26, 30] and flexible job shop [4, 5, 12, 13, 22, 32]. Formulations with equal [5, 6, 7, 15, 26, 29, 31, 35, 36] or consistent sublots [7, 11, 13, 16, 22, 24, 26, 31] are most often considered. The more complex lot streaming problems with variable sublots are studied rarely

[2, 10, 25, 28, 30], in particular, only a single work is known to the authors which concerns the FJS lot streaming with variable sublots [4].

Lot sizing is a general term denoting optimization problems in which sizes of production lots are determined. The classification of lot sizing problems is based on capacity constraints, the planning horizon, the number of levels, the number of products, and others [23]. The optimization of sublot sizes in scheduling problems with lot streaming is referred to as lot sizing as well, and this meaning is used in this work. In some of the previously mentioned works, lot streaming is complemented by lot sizing, usually in the case of the FS problems [2, 11, 16, 25, 28]. The JS lot streaming problem is also studied in combination with lot sizing, but equal or consistent sublots are considered [6, 7], and only one case of variable sublots [30] with split of jobs into 2 or 3 sublots. Lot sizing with consistent sublots has been also applied to FJS [22].

Because of practical pragmatic requirements, formulations of the problems are often complemented by additional extensions, regardless of lot streaming and lot sizing are used or not. There are models with blocking [19], limited capacity buffers [36], products assembly representation [7], preventive maintenance [28], and calendars of machine availability [4]. However, transport times [12, 15, 19, 24, 33] and setup times [10, 12, 19, 20, 21, 29, 32, 33] are the most frequently considered extensions. Because of that, the transport times and sequence-dependent setup times have been also included into the problem formulation in this work.

The problems with lot streaming and optionally with lot sizing are typically modeled with the use of a mathematical programming formalism, especially mixed-integer linear programming (MILP) [2, 28, 32]. Even if other methods are used for optimization, a mathematical model is often the starting point. The set of used optimization approaches includes: genetic algorithms (GA) [6, 7, 10, 11, 12, 13, 30, 32, 36, 37], tabu search (TS) [4, 16, 19], particle swarm optimization (PSO) [22, 32, 35], dispatching rules [5, 7, 15], and others. The vast majority of research related to lot streaming and lot sizing in FJS use the makespan as the objective function. Only a few works consider different objectives [5, 7, 15, 35], for instance, average tardiness, weighted earliness and tardiness, inventory cost, flow time, number of tardy jobs.

The authors noticed that there is relatively small number of works devoted to the FS with lot sizing [7] and that the FJS with lot streaming is complex for modeling and optimization [12, 13]. On the other hand, the results of previous research indicate that lot streaming with variable lots is needed to effectively optimize the underlying objective, such as the makespan, using lot sizing [25]. There have probably been no approaches combining the FJS and lot sizing with variable sublots, yet. The main goal of the present work is to fill this gap.

3 Problem formulation

The problem considered can be described as follows:

- A set of n jobs $\mathbf{J} = \{J_k : k \in \{1, \dots, n\}\}$ is given and a parameter $e_k \in \mathbb{R}_+$ (positive real numbers) represents the size of the job J_k.
- A set of operations $\mathbf{O}_k = \{O_{k,i} : i \in \{1, \dots, o(k)\}\}$ is defined for each job J_k, where $O_{k,i}$ denotes the i-th operation of the job and $o(k) = |\mathbf{O}_k|$. The set $\mathbf{O} = \bigcup_{k \in \{1,\dots,n\}} \mathbf{O}_k$ contains all operations of the problem instance.
- A set of r machines $\mathbf{M} = \{M_p : p \in \{1, \dots, r\}\}$ is given.
- A minimal sublot size $d_{k,i} \in \mathbb{R}_+$, $0 < d_{k,i} \leq e_k$ is defined for each operation $O_{k,i}$. In the lot sizing decision process an actual size $a_{k,i} \in [d_{k,i}, e_k]$ has to be assigned to each operation.

As the result, the operation will be split into $\lceil e_k / a_{k,i} \rceil$ sublots, with each sublot of size $a_{k,i}$, except the last one, possibly having a smaller size equal to $e_k - (\lceil e_k / a_{k,i} \rceil - 1)\, a_{k,i}$. The function $g(k,i) = \lceil e_k / d_{k,i} \rceil$ returns an upper bound on the number of sublots of an operation $O_{k,i}$.

- A non-empty set $\mathbf{M}_{k,i} \in \mathbf{M}$ is assigned to each operation which includes the machines that can process sublots of this operation.
- If an operation $O_{k,i}$ can be processed on machine M_p, a unit processing time $p_{k,i,p} \in \mathbb{R}_+$ is defined. Therefore, the effective processing time of a sublot of size $a_{k,i}$ on the given machine equals $a_{k,i}\, p_{k,i,p}$.
- The setup time needed for a machine M_p to changeover from processing an operation O_{k_1,i_1} to an operation O_{k_2,i_2} is represented by the parameter $s_{p,k_1,i_1,k_2,i_2} \in \mathbb{R}_+ \cup \{0\}$.
- The time of transport from a machine M_{p_1} to M_{p_2} is defined as $t_{p_1,p_2} \in \mathbb{R}_+ \cup \{0\}$.
- Batch processing of sublots is presupposed. This means that all sublot volume has to be delivered to a target machine to start processing, and all the volume becomes available for consecutive operations after the elapse of the processing time.
- The machines can execute at most one setup or processing activity at once.
- The transport, setup and processing activities related to a single sublot have to be executed without interruption.
- A two-level lexicographic optimization procedure is defined with a scheduling stage followed by lot sizing:

$$\begin{array}{llll} \text{first find:} & \mathcal{S}^* \in \mathbb{S} & \text{s.t.} & \left[\underset{O_{k,i} \in \mathbf{O}}{\forall} a_{k,i} = d_{k,i} \right] \wedge f_{\text{sched}}(\mathcal{S}^*) \longrightarrow \min! \\ \text{then find:} & \mathcal{S}^{**} \in \mathbb{S} & \text{s.t.} & f_{\text{sched}}(\mathcal{S}^{**}) \leq f_{\text{sched}}(\mathcal{S}^*) \wedge f_{\text{size}}(\mathcal{S}^{**}) \longrightarrow \max!\,, \end{array}$$

where $\mathbb{S}$ is the space of all feasible solutions, $\mathcal{S}^*$ is an intermediate solution optimized with respect to the scheduling objective, and $\mathcal{S}^{**}$ is the final solution of the problem. Having a solution $\mathcal{S} \in \mathbb{S}$, the function $\mathcal{C} : \mathbb{S} \times \mathbf{J} \to \mathbb{R}_+$ can be defined which assigns the completion time $c_k = \mathcal{C}(\mathcal{S}, J_k)$ to each job $J_k \in \mathbf{J}$ in this solution. The function $f_{\text{sched}} : \mathbb{S} \to \mathbb{R}_+$, minimized in the first stage of the optimization, represents the makespan of a schedule $\mathcal{S}$:

$$f_{\text{sched}}(\mathcal{S}) = \text{makespan}(\mathcal{S}) = \max_{J_k \in \mathbf{J}} \mathcal{C}(\mathcal{S}, J_k)\ .$$

The second stage of the optimization is considered in two variants defined by the following functions:

$$f^{\text{A}}_{\text{size}}(\mathcal{S}) = \sum_{O_{k,i} \in \mathbf{O}} a_{k,i}\,, \quad f^{\text{B}}_{\text{size}}(\mathcal{S}) = \sum_{\substack{O_{k,i} \in \mathbf{O} \\ \text{s.t. } a_{k,i} = e_k}} 1\,.$$

The functions $f^{\text{A}}_{\text{size}}$ and $f^{\text{B}}_{\text{size}}$ represent the sum of the sublot sizes and the number of non-split lots, respectively.

4 Graph model

The minimization of makespan is a regular optimization criterion, therefore an optimal solution belongs to the set of left-shifted schedules and can be unambiguously represented by a pair (Γ, Π), where $\Gamma = \{a_{k,i}\}$ is the set of lot sizes assigned to the operations $O_{k,i} \in \mathbf{O}$ and $\Pi = \{\pi_p\}$ is the set of processing orders on the machines $M_p \in \mathbf{M}$. Each processing order $\pi_p \in \Pi$ is a sequence of the operations $\pi_p = \Big(\pi_{p,1}, \pi_{p,2}, \ldots, \pi_{p,z(p)}\Big)$ processed on the machine M_p one by one, where $\pi_{p,q} \in \mathbf{O}$ and $z(p)$ is the number of jobs processed on this machine. The elements $\pi_{p,q}$ are subject to the following constraints:

$$\underset{\pi_{p_1,q_1},\pi_{p_2,q_2}\in\mathbf{O}}{\forall} p_1 \neq p_2 \vee q_1 \neq q_2 \implies \pi_{p_1,q_1} \neq \pi_{p_2,q_2}\,,$$

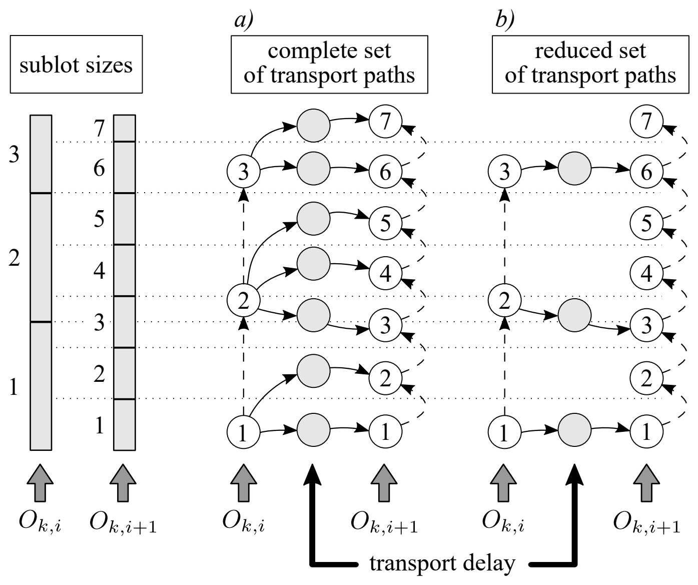
$$\underset{\pi_{p,q}\in\mathbf{O}}{\forall}\ \underset{O_{k,i}\in\mathbf{O}}{\exists} \pi_{p,q} = O_{k,i} \wedge M_p \in \mathbf{M}_{k,i}\,, \qquad \bigcup_{\substack{p\in\{1,\ldots,r\}\\ q\in\{1,\ldots,z(p)\}}} \{\pi_{p,q}\} = \mathbf{O}\,.$$

For a given pair (Γ, Π), a directed vertex-weighted graph $\mathbf{G}(\Gamma, \Pi)$ modeling the related left-shifted schedule can be constructed in which the vertices represent the activities of transport, setup and processing of sublots, while the arcs represent the precedence relations between the activities. In comparison to the graph model of the standard job shop problem, in the considered model a specific representation of sublots and relationships between them are used. Let us consider a pair of consecutive operations $(O_{k,i}, O_{k,i+1})$ of a single job J_k. The job size equals e_k and the sublot sizes of the operations are equal to $a_{k,i}$ and $a_{k,i+1}$, respectively. The u-th sublot in the processing sequence of an operation $O_{k,i}$ will be denoted by $O_{k,i,u}$, $u \in \{1, \ldots, g(k,i)\}$. As a consequence of the quantity balance of the good processed, the processing of the sublot $O_{k,i+1,v}$ can start after completing the processing of the sublot $O_{k,i,u}$ if the condition $u\,a_{k,i} \geq v\,a_{k,i+1}$ is satisfied, except the last sublot $O_{k,i+1,g(k,i+1)}$ which can start after the last sublot $O_{k,i,g(k,i)}$ of the previous operation. As the result, the sublot $O_{k,i+1,v}$ is preceded by the sublot $O_{k,i,u}$ on a transport path, where $u = \min\left(\lceil v\,a_{k,i+1}/a_{k,i}\rceil, g(k,i)\right)$. The exemplary transport paths are shown in Figure 2a for the case $a_{k,i} > a_{k,i+1}$. It can be additionally noticed that some of these paths are redundant in the context of the longest path determination, because all transport times between the same pair of machines are equal. For example, the direct path from $O_{k,i,2}$ to $O_{k,i+1,5}$ is redundant (Fig. 2a), because the indirect path through the vertices $O_{k,i,2}$, $O_{k,i+1,3}$, $O_{k,i+1,4}$, and $O_{k,i+1,5}$ has to be longer. The transport connections after removing the redundant paths are presented in Figure 2b. In general, a transport path between the vertices $O_{k,i,u}$ and $O_{k,i+1,v+1}$ is redundant if there exists a transport path from $O_{k,i,u}$ to $O_{k,i+1,v}$. Finally, summarizing the mentioned observations, the function

$$\mathcal{T}(k,i,r) = \Big\{ (u,v) : v \in \{1,\ldots,g(k,i+1)\} \\ \wedge\ u = \min\Big(\lceil v\,r\rceil\,, g(k,i)\Big) \ \wedge\ u \neq \lceil (v-1)\,r\rceil \Big\} \tag{1}$$

can be defined which assigns the set of index pairs (u,v) of the sublots connected by transport paths between consecutive operations $O_{k,i}$ and $O_{k,i+1}$ for a given $r = a_{k,i+1}/a_{k,i}$. The function $\mathcal{T}$ also describes how to derive the transport paths in an algorithmic way.

The complete solution graph $\mathbf{G}(\Gamma, \Pi) = (\mathrm{V}, \mathrm{A}, w)$ is defined by a set of vertices V, a set of



a)
b)
sublot sizes
complete set of transport paths
reduced set of transport paths
$O_{k,i}$
$O_{k,i+1}$
transport delay

Figure 2: Exemplary subgraphs of $\mathbf{G}(\Gamma,\Pi)$ modeling transport paths

arcs A, and a weight function w as follows:

$$\mathrm{V}=\bigcup_{i=1}^{4}\mathrm{V}_i\,,\quad \mathrm{V}_1=\{V_\mathrm{s},V_\mathrm{c}\}\,,\quad \mathrm{V}_2=\bigcup_{k=1}^{n}\bigcup_{i=1}^{o(k)}\bigcup_{b=1}^{g(k,i)}\left\{V^\mathrm{P}_{k,i,b}\right\}, \tag{2}$$

$$\mathrm{V}_3=\bigcup_{k=1}^{n}\bigcup_{i=1}^{o(k)-1}\bigcup_{(u,v)\in\mathcal{T}(k,i)}\left\{V^\mathrm{T}_{k,i,u,v}\right\},\quad \mathrm{V}_4=\bigcup_{p=1}^{r}\bigcup_{q=1}^{z(p)-1}\left\{V^\mathrm{S}_{q,p}\right\}, \tag{3}$$

$$\mathrm{A}=\bigcup_{i=1}^{4}\mathrm{A}_i\,,\quad \mathrm{A}_1=\bigcup_{k=1}^{n}\left[\left\{\left(V_\mathrm{s},V^\mathrm{P}_{k,1,1}\right)\right\}\cup\left\{\left(V^\mathrm{P}_{k,o(k),g(k,i)},V_\mathrm{c}\right)\right\}\right], \tag{4}$$

$$\mathrm{A}_2=\bigcup_{k=1}^{n}\bigcup_{i=1}^{o(k)-1}\bigcup_{(u,v)\in\mathcal{T}(k,i)}\left[\left\{\left(V^\mathrm{P}_{k,i,u},V^\mathrm{T}_{k,i,u,v}\right)\right\}\cup\left\{\left(V^\mathrm{T}_{k,i,u,v},V^\mathrm{P}_{k,i+1,v}\right)\right\}\right], \tag{5}$$

$$\mathrm{A}_3=\bigcup_{p=1}^{r}\bigcup_{q=1}^{z(p)}\bigcup_{b=1}^{g(k,i)-1}\left\{\left(V^\mathrm{P}_{k,i,b},V^\mathrm{P}_{k,i,b+1}\right)\right\},\qquad (p,q,k,i):\ \pi_{p,q}=O_{k,i}, \tag{6}$$

$$\mathrm{A}_4=\bigcup_{p=1}^{r}\bigcup_{q=1}^{z(p)-1}\left[\left\{\left(V^\mathrm{P}_{k_1,i_1,g(k_1,i_1)},V^\mathrm{S}_{q,p}\right)\right\}\cup\left\{\left(V^\mathrm{S}_{q,p},V^\mathrm{P}_{k_2,i_2,1}\right)\right\}\right],$$
$$(p,q,k_1,i_1,k_2,i_2):\ \pi_{p,q}=O_{k_1,i_1},\ \pi_{p,q+1}=O_{k_2,i_2}, \tag{7}$$

$$\underset{\substack{O_{k,i}\in\mathbf{O}\\ b\in g(k,i)}}{\forall}\ w\left(V^\mathrm{P}_{k,i,b}\right)=p_{k,i,p}\min\left(a_{k,i},\ e_k-(b-1)a_{k,i}\right), \tag{8}$$

$$\underset{\substack{O_{k,i}\in\mathbf{O}\\ (u,v)\in\mathcal{T}(k,i)}}{\forall}\ w\left(V^\mathrm{T}_{k,i,u,v}\right)=t_{p_1,p_2},\quad (p_1,p_2,k,i):\ \pi_{p_1,q_1}=O_{k,i},\ \pi_{p_2,q_2}=O_{k,i+1}, \tag{9}$$

$$\underset{\substack{p\in\{1,\dots,r\}\\ q\in\{1,\dots,z(p)-1\}}}{\forall}\ w\left(V^\mathrm{S}_{p,q}\right)=s_{p,k_1,i_1,k_2,i_2},\quad (p,q,k_1,i_1,k_2,i_2):\ \pi_{p,q}=O_{k_1,i_1},\ \pi_{p,q+1}=O_{k_2,i_2}, \tag{10}$$

$$w\left(V_\mathrm{s}\right)=w\left(V_\mathrm{c}\right)=0\,. \tag{11}$$

The subset V_1 consists of a source (V_s) and a sink (V_c) vertex of the graph. The vertices $V^\mathrm{P}_{k,i,b}\in\mathrm{V}_2$ represent the processing activities of the related sublots $O_{k,i,b}$, the vertices $V^\mathrm{T}_{k,i,u,v}\in\mathrm{V}_3$

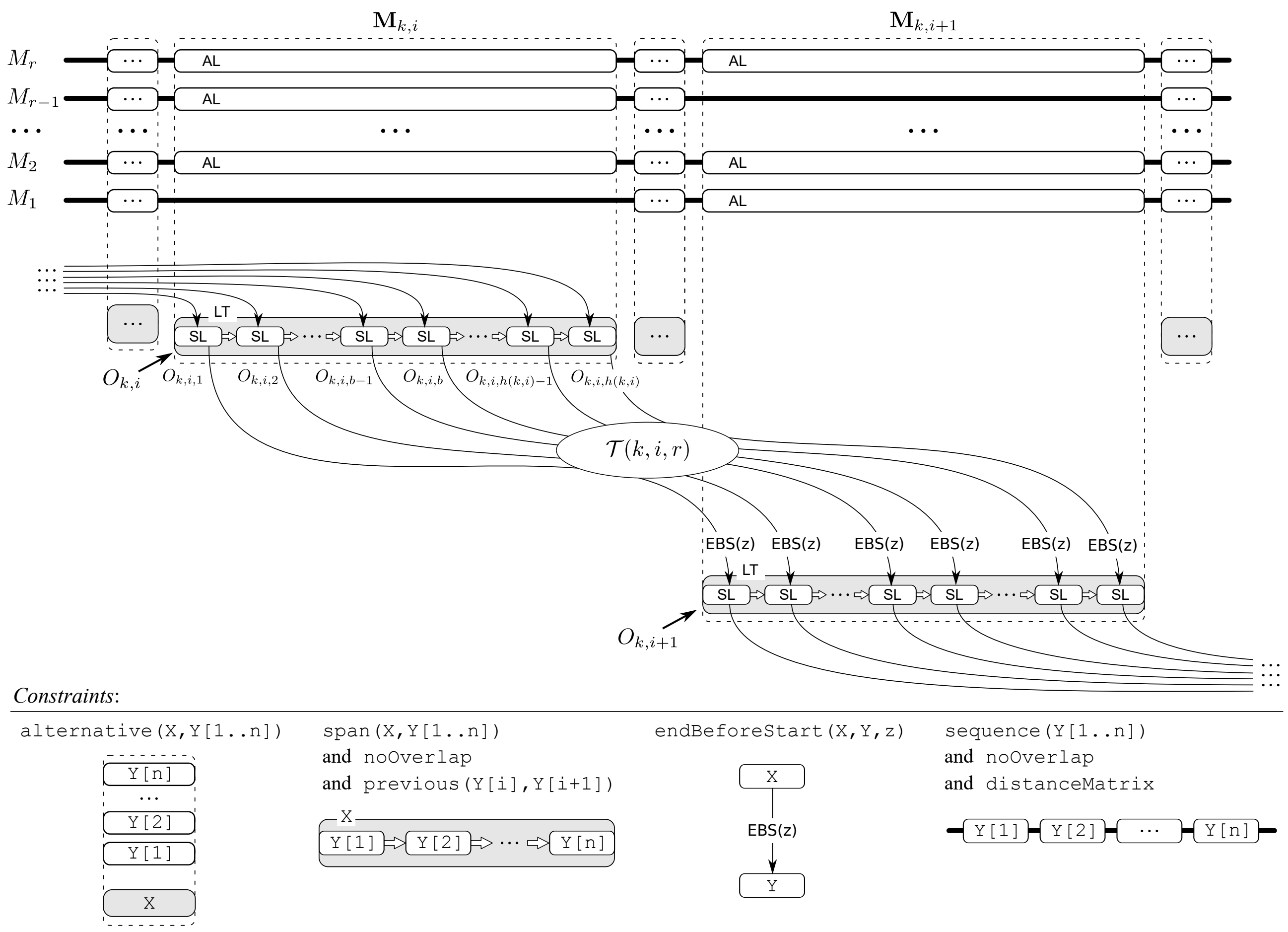


Figure 3: Constraint programming model for the scheduling problem

model transport delays between the sublots $O_{k,i,u}$ and $O_{k,i+1,v}$, whereas the vertices $V^{\mathrm{S}}_{p,q} \in \mathrm{V}_4$ map setup activities between the last sublots of the operations $\pi_{p,q}$ and the first sublots of the operations $\pi_{p,q+1}$. The arcs from the set A_1 connect appropriate vertices with the source or sink, the arcs from A_2 constitute transport paths, the arcs from A_3 impose a processing order for consecutive sublots of operations, and the arcs from A_4 connect successive operations on a single machine by paths with intermediate setup vertices.

5 Scheduling

5.1 Constraint programming

The set of extensions of a standard constraint programming provided by IBM ILOG CP Optimizer (CPO) for modeling and finding optimal solutions of scheduling problems makes it possible to represent a problem in a structural form using high-level decision variables (*interval variables* and *sequence variables*) and dedicated types of constraints. The interval variables represent the scheduled activities and their properties such as start time, end time, size, presence, etc. The sequence variables model permutations of some sets of interval variables. The structure of the model implemented for the considered problem is shown in Figure 3. Three kinds of interval variables have been defined. Each sublot is modeled by a single interval variable SL, entire operations (lots) are mapped by the variables LT, and for each optional pair (operation, machine) a variable AL is defined. Each variable LT representing an operation $O_{k,i} \in \mathbf{O}$ is bound with the variables SL modeling the related sublots $O_{k,i,u}$, $u \in \{1,\dots,g(k,i)\}$, by the constraint `span`. This constraint enforces that the start and end times of the LT variable have to be equal to the earliest start time and latest end time of the variables from the related set SL, respectively. Additional constraints (`no overlap`, `previous`) impose precedence relations on the variables SL modeling consecutive sublots $O_{k,i,b}$, $O_{k,i,b+1}$ of the same operation, these constraints are exactly equivalent to the arcs A_3 (6) from the graph model. The transport paths are modeled by the constraints `endBeforeStart(X,Y,t)` defined for each pair of variables SL related to the sublots $O_{k,i,u}, O_{k,i+1,v}$, according to the function $\mathcal{T}(k,i,r)$. The parameter `t` denotes the minimal delay required between the end time of `X` and the start of `Y`, thus it represents the transport time. A constraint `alternative` is imposed on each variable LT related to an operation $O_{k,i}$ and the subset of variables AL defined for optional machines $M_p \in \mathbf{M}_{k,i}$ which can process this operation. This constraint assures that exactly one variable AL from the subset has to be present (the others are excluded from the schedule), additionally the start and end times of the variable LT and the selected variable AL have to be equal. As the result, according to the problem formulation, a single machine will be chosen for each operation $O_{k,i} \in \mathbf{O}$ and this machine will be occupied by the operation from the start of the first sublot $O_{k,i,1}$ to the end of the last sublot $O_{k,i,g(k,i)}$. For each operation $O_{k,i} \in \mathbf{O}$, an integer decision variable is defined and coupled with the respective `alternative` constraint in such a way that the value of this variable indicates which machine has been chosen for the operation. These variables are used in decision expressions which determine machine-dependent parameters, i.e., the processing times modeled by the sizes of the variables SL and the transport times represented by the parameter `t` in the constraints `endBeforeStart(X,Y,t)`. All the variables AL related to the same machine are covered by a single sequence variable, thus any permutation of these variables can appear in a solution. Additionally, constraints `no overlap` are imposed on these sequence variables, and the required setup times are introduced using the parameter `distance matrix` which defines minimal time gaps between any two contained interval variables.

5.2 Tabu search

The second implemented scheduling algorithm is based on a tabu search metaheuristic. The framework of the algorithm is represented by Pseudocode 1. The calculation of the objective function value has been parallelized to exploit as much computational power of a modern multicore CPU as possible. All solutions belonging to a particular neighborhood are added to a parallel computational queue and the resulting values of the objective function are received by the main thread after the computations.

The standard aspiration criterion is used in the algorithm such that a new solution is accepted if it is better than the current global best solution, even though it is forbidden by the tabu list.

The makespan value is calculated as the length of a longest path from V_s to V_c in the solution graph $\mathbf{G}(\Gamma,\Pi)$ with time complexity $O(\mu)$, where $\mu = \sum_{O_{k,i}\in\mathbf{O}} \lceil e_k/d_{k,i} \rceil$ is the number of all sublots in the schedule.

The neighborhood of a solution is generated as follows:

- All critical operations of the solution are found. An operation is classified to be critical if at least one of its sublot is critical, and a sublot is critical if it is represented by a vertex of $\mathbf{G}(\Gamma,\Pi)$ which belongs to a longest path from V_s to V_c.

- New solutions are constructed by moving all the critical operations (only one operation at once) to all possible new positions of the processing queue on the same or another available machine. The neighborhood is the set of all these solutions.

Pseudocode 1: Tabu search framework

```
globalBestSolution := bestSolution := getInitialSolution();
globalBestObjective := calculateObjective(globalBestSolution);
while not stopCondition.isSatisfied() do
    foreach move in neighborhood do
        computationalQueue.addTask(bestSolution, move);
    end
    bestSolution := null;
    bestObjective := $\infty$;
    while not computationalQueue.isEmpty() do
        (solution, objective) := computationalQueue.nextResult();
        if solution.isFeasible() then
            tabu := tabuList.contains(getTabuAttribute(solution));
            if (objective < bestObjective and not tabu)
                    or (objective < globalBestObjective) then
                bestSolution := solution;
                bestObjective := objective;
                if bestObjective < globalBestObjective then
                    globalBestSolution := bestSolution;
                    globalBestObjective := bestObjective;
                end
            end
        end
    end
    if bestSolution = null then
        exitWithError("No feasible solutions in neighborhood");
    end
    tabuList.add(getTabuAttribute(bestSolution));
end
```

The tabu attribute has been defined as the sum of the numbers of all sublots belonging to the critical path of the related solution. Unique numbers are assigned to sublots before the main optimization algorithm starts. Such a tabu attribute has transpired to be the most effective one among a few tested formulations, i.e., the objective function is minimized fastest and the algorithm does not have an inclination to stall in cyclic trajectories of the search space.

The length of the tabu list is adjusted dynamically, using a rule similar to that presented in [3]. The length is given by the parameter

$$tl = \left\lfloor \frac{\sum_{i=1}^{tl^*} fmc_{(-i)}}{tl^* \, tl^{\mathrm{R}}} \right\rfloor, \tag{12}$$

where tl^* is the previous length of the tabu list, $fmc_{(-i)}$ (*feasible move counter*) denotes the number of feasible solutions from the i-th iteration of the algorithm backwards, and tl^{R} is a constant experimentally tuned parameter. The calculated length tl is also the number of iterations of the algorithm in which this length is in force, then, a new length is derived from expression (12) again. The initial length of the tabu list, denoted by tl^{I}, is a parameter of the algorithm.

The algorithm stops when the following condition is satisfied:

$$\left(t \geq t^{\mathrm{MIN}} \wedge \frac{t^{\mathrm{LI}\rightarrow}}{t^{\rightarrow\mathrm{LI}}} \geq \mathit{stag}\right) \vee \mathit{stop}, \tag{13}$$

where t is the actual running time of the algorithm, t^{MIN} denotes a lower bound on the running time, $t^{\rightarrow\mathrm{LI}}$ and $t^{\mathrm{LI}\rightarrow}$ are times elapsed before and after the last improvement of a solution, respectively, *stag* is a stagnation coefficient, and *stop* is the flag of an external stop request.

6 Lot sizing

6.1 Maximization of the sum of the sublot sizes

6.1.1 MILP optimization

The problem is characterized by the following decision variables:

$\alpha_{k,i} \in [d_{k,i}, e_k]$	– sublot size assigned to the operation $O_{k,i}$,
$\sigma_{k,i,b} \in [0, e_k]$	– individual size of the sublot $O_{k,i,b}$,
$\gamma_{k,i,b} \in [0, \mathcal{M}]$	– start time of the sublot $O_{k,i,b}$,
$o_{k,i,b} \in \{0,1\}$	– if equal to 0, the condition $\sigma_{k,i,b} = 0$ has to be satisfied,
$\iota_{k,i,b} \in \{0,1\}$	– if equal to 0, the condition $\sigma_{k,i,b} = \alpha_{k,i}$ has to be satisfied,
$\tau_{k,i,u,v} \in \{0,1\}$	– if equal to 0, a transport connection exists from the sublot $O_{k,i,u+1}$ to the sublot $O_{k,i+1,v}$,

where

$$\begin{array}{lll} (k,i,u,v): & \begin{array}{l} k \in \{1,\dots,n\}\,,\ i \in \{1,\dots,o(k)-1\}\,, \\ u \in \{1,\dots,g(k,i)-1\}\,,\ v \in \{1,\dots,g(k,i+1)\} \end{array} & \text{for } \tau_{k,i,u,v}\,, \\ (k,i,b): & O_{k,i} \in \mathbf{O},\ b \in \{1,\dots,g(k,i)\} & \text{for the other variables,} \end{array} \tag{14}$$

and $\mathcal{M}$ is the makespan value obtained in the previous stage of optimization.

The lot sizing problem is described in the form of an MILP problem as follows:

$$\sum_{O_{k,i} \in \mathbf{O}} \alpha_{k,i} \longrightarrow \max!$$

subject to

$$\sigma_{k,i,b} \leq \alpha_{k,i}\,, \qquad (k,i,b):\ O_{k,i} \in \mathbf{O},\ b \in \{1,\dots,g(k,i)\}\,, \tag{15}$$

$$\sigma_{k,i,b} \geq \alpha_{k,i} - e_k\, \iota_{k,i,b}\,, \qquad (k,i,b):\ O_{k,i} \in \mathbf{O},\ b \in \{1,\dots,g(k,i)\}\,, \tag{16}$$

$$\sigma_{k,i,b} \leq e_k\, o_{k,i,b}\,, \qquad (k,i,b):\ O_{k,i} \in \mathbf{O},\ b \in \{1,\dots,g(k,i)\}\,, \tag{17}$$

$$\iota_{k,i,b} \leq \iota_{k,i,b+1}\,, \qquad (k,i,b):\ O_{k,i} \in \mathbf{O},\ b \in \{1,\dots,g(k,i)-1\}\,, \tag{18}$$

$$o_{k,i,b} \geq o_{k,i,b+1}\,, \qquad (k,i,b):\ O_{k,i} \in \mathbf{O},\ b \in \{1,\dots,g(k,i)-1\}\,, \tag{19}$$

$$z \leq \sum_{b=1}^{z} (o_{k,i,b} + \iota_{k,i,b}) \leq z + 1, \quad (k,i,z) : \, O_{k,i} \in \mathbf{O}, \; z = g(k,i), \tag{20}$$

$$\sum_{b=1}^{u} \sigma_{k,i,b} \geq \sum_{b=1}^{v} \sigma_{k,i+1,b} + (\tau_{k,i,u,v} - 1) e_k \,, \quad \begin{array}{l} (k,i,u,v) : \, k \in \{1,\ldots,n\} \,, \, i \in \{1,\ldots,o(k)-1\} \,, \\ u \in \{1,\ldots,g(k,i)-1\} \,, \, v \in \{1,\ldots,g(k,i+1)\} \,, \end{array} \tag{21}$$

$$\mathcal{M}\tau_{k,i,u,v} + \gamma_{k,i+1,v} \geq \gamma_{k,i,u+1} + \sigma_{k,i,u+1} p_{k,i,p_1} + t_{p_1,p_2} \,, \quad \begin{array}{l} (k,i,u,v,p_1,p_2) : k \in \{1,\ldots,n\} \,, i \in \{1,\ldots,o(k)-1\} \,, \\ u \in \{1,\ldots,g(k,i)-1\} \,, \, v \in \{1,\ldots,g(k,i+1)\} \,, \\ \pi_{p_1,q_1} = O_{k,i} \in \mathbf{O}_k, \; \pi_{p_2,q_2} = O_{k,i+1} \in \mathbf{O}_k \,, \end{array} \tag{22}$$

$$\gamma_{k,i+1,1} \geq \gamma_{k,i,1} + \sigma_{k,i,1} p_{k,i,p_1} + t_{p_1,p_2} \,, \quad \begin{array}{l} (k,i,p_1,p_2) : \, k \in \{1,\ldots,n\} \,, \, i \in \{1,\ldots,o(k)-1\} \,, \\ \pi_{p_1,q_1} = O_{k,i} \in \mathbf{O}_k, \; \pi_{p_2,q_2} = O_{k,i+1} \in \mathbf{O}_k \,, \end{array} \tag{23}$$

$$\gamma_{k,i+1,g(k,i+1))} \geq \gamma_{k,i,g(k,i)} + \sigma_{k,i,g(k,i)} p_{k,i,p_1} + t_{p_1,p_2} \,, \quad \begin{array}{l} (k,i,p_1,p_2) : \, k \in \{1,\ldots,n\} \,, \, i \in \{1,\ldots,o(k)-1\} \,, \\ \pi_{p_1,q_1} = O_{k,i} \in \mathbf{O}_k, \; \pi_{p_2,q_2} = O_{k,i+1} \in \mathbf{O}_k \,, \end{array} \tag{24}$$

$$\gamma_{k,i,b+1} \geq \gamma_{k,i,b} + \sigma_{k,i,b} p_{k,i,p} \,, \quad \begin{array}{l} (k,i,p,b) : \, p \in \{1,\ldots,r\} \,, \, q \in \{1,\ldots,z(p)\} \,, \\ O_{k,i} = \pi_{p,q} \in \mathbf{O}, \; b \in \{1,\ldots,g(k,i)-1\} \,, \end{array} \tag{25}$$

$$\gamma_{k_2,i_2,1} \geq \sigma_{k_1,i_1,g(k_1,i_1)} p_{k_1,i_1,p} + s_{p,k_1,i_1,k_2,i_2} + \gamma_{k_1,i_1,g(k_1,i_1)} \,, \quad \begin{array}{l} (k_1,i_1,k_2,i_2,p) : \, p \in \{1,\ldots,r\} \,, \, q \in \{1,\ldots,z(p)-1\} \,, \\ O_{k_1,i_1} = \pi_{p,q} \in \mathbf{O}, \; O_{k_2,i_2} = \pi_{p,q+1} \in \mathbf{O} \,, \end{array} \tag{26}$$

$$\gamma_{k,i,g(k,i)} + \sigma_{k,i,g(k,i)} p_{k,i,p} \leq \mathcal{M} \,, \quad (k,i,p) : \, p \in \{1,\ldots,r\} \,, \, q = z(p), \; O_{k,i} = \pi_{p,q} \,. \tag{27}$$

In the given MILP model, the variables $\sigma_{k,i,b}$ and $\gamma_{k,i,b}$ have to be defined for the maximal possible number of sublots $b \in \{1,\ldots,g(k,i)\}$, because the actual number is a function of $\alpha_{k,i}$ and it is not known in advance. Therefore, in the sequence of the sublot sizes $\left(\sigma_{k,i,1}, \sigma_{k,i,2}, \ldots, \sigma_{k,i,g(k,i)}\right)$, there are full sublots first, such that $\sigma_{k,i,b} = \alpha_{k,i}$, then possibly a single not full sublot, such that $\sigma_{k,i,b} \in (0, \alpha_{k,i})$, and possibly empty surplus sublots at the end, such that $\sigma_{k,i,b} = 0$. The sublot sizes are controlled by the constraints (15)-(20). The size of an individual sublot $\sigma_{k,i,b}$ has to be not greater than the general size of sublots $\alpha_{k,i}$ assigned to an operation (15), moreover the equality $\sigma_{k,i,b} = \alpha_{k,i}$ (a full sublot) is imposed if $\iota_{k,i,b} = 0$ (16), whereas the condition $\sigma_{k,i,b} = 0$ (a surplus sublot) has to be satisfied if $o_{k,i,b} = 0$ (17). The value e_k is used as a *large enough constant* to deactivate the constraints (16) or (17) if $\iota_{k,i,b} = 1$ or $o_{k,i,b} = 1$, respectively. The constraints (18) enforce that, if a sublot $O_{k,i,b}$ is full, the previous one $O_{k,i,b-1}$ will also be full. Analogously, the constraints (19) ensure that, if a sublot $O_{k,i,b}$ is empty, the next one $O_{k,i,b+1}$ will be empty as well. The problem formulation permits only one sublot of an operation to be not full (and not empty), and this condition is guaranteed by the constraints (20).

As the sublot sizes are not known in advance, the transport paths cannot be fixed using the input data and have to be determined dynamically in the optimization process. The relevant constraints are derived from the observation that, if the cumulative size of the sublots $O_{k,i,b}$, $b \in \{1,2,\ldots,u\}$, is less than the cumulative size of sublots of the consecutive operation $O_{k,i+1,b}$, $b \in \{1,2,\ldots,v\}$, then the sublot $O_{k,i+1,v}$ cannot start after the completion time of $O_{k,i,u}$ increased by the transport delay, but at least the sublot $O_{k,i,u+1}$ has to be additionally processed before. The given condition is implemented by the constraints (21), which compare the cumulative sublot sizes, and by (22), which impose required time relations between the sublot start times. These constraints are coupled by the binary variables $\tau_{k,i,u,v}$, whereas the values e_k and $\mathcal{M}$ are used as *large enough constants*. In the constraints (22), the elements $\sigma_{k,i,u+1}\, p_{k,i,p_1}$ and t_{p_1,p_2} represent the processing time of the sublot $O_{k,i,u+1}$ and the transport time between machines, respectively, needed for imposing a proper start-to-start time relation. Additionally, the transport delays have to be always present between the first and the last sublots of consecutive operations, it is not covered by (21) and (22), but introduced in separate constraints (23) and (24), respectively.

The constraints (25) ensure that the start time of each sublot $O_{k,i,b+1}$ is not less than the completion time of the previous one $O_{k,i,b}$. The start time of the first sublot of an operation has also to be not less than the completion time of the last sublot of the previous operation on the same machine increased by the relevant setup time (26). The constraints (27) assure that the completion times of the last sublots processed on the respective machines do not exceed the demanded makespan value.

6.1.2 Discrete search with linear programming

The considered lot sizing problem can be represented using a linear programming model without integer variables in any part of the search space for which the following conditions are satisfied:

- the number of sublots of each operation remains constant,
- the configuration of all transport paths is fixed.

Let us assume that the number of sublots of each operation $O_{k,i} \in \mathbf{O}$ is fixed and equal to $\eta_{k,i}$. The last sublot of the operation can be almost empty, then the sublot size is close to $e_k/(\eta_{k,i}-1)$ from below. When the last sublot is full, in turn, the size becomes equal to $e_k/\eta_{k,i}$. Thus, if the sublot number is fixed at $\eta_{k,i}$, the sublot size $a_{k,i}$ is limited as follows:

$$a_{k,i} \in \mathbf{a}(\eta_{k,i}) = \begin{cases} \left[d_{k,i}, \frac{e_k}{\eta_{k,i}-1}\right), & \text{for} \quad \eta_{k,i} = g(k,i), \\ \left[\frac{e_k}{\eta_{k,i}}, \frac{e_k}{\eta_{k,i}-1}\right), & \text{for} \quad \eta_{k,i} \in \{2, \ldots, g(k,i)-1\}, \\ e_k, & \text{for} \quad \eta_{k,i} = 1. \end{cases} \tag{28}$$

It is easy to see on the basis of Figure 2 and relation (1) that the configuration of transport paths changes at the values of the sublot sizes $a_{k,i}, a_{k,i+1}$ of operations $O_{k,i}, O_{k,i+1} \in \mathbf{O}$ having the proportions $\kappa_{k,i+1,u}/\kappa_{k,i,u} = a_{k,i+1}/a_{k,i}$, where $\kappa_{k,i+1,u}/\kappa_{k,i,u} \in \mathbf{K}_{k,i}$, $u \in \{1, \ldots, |\mathbf{K}_{k,i}|\}$. The set $\mathbf{K}_{k,i}$ includes all rational numbers for which the numerator and denominator are integers not greater than the number of full sublots of the related operations in swapped order, i.e., $\kappa_{k,i+1,u} \in \{1, \ldots, \eta_{k,i}-1\}$, $\kappa_{k,i,u} \in \{1, \ldots, \eta_{k,i+1}-1\}$ for $\kappa_{k,i+1,u}/\kappa_{k,i,u} \in \mathbf{K}_{k,i}$ (the last sublot is considered to be not full). Additionally, the order is imposed such that $\kappa_{k,i+1,u-1}/\kappa_{k,i,u-1} < \kappa_{k,i+1,u}/\kappa_{k,i,u}$. If the numbers $\eta_{k,i}, \eta_{k,i+1}$ are fixed at the values specified in the second variant of expression (28), the ratio $a_{k,i+1}/a_{k,i}$ has to be greater than $(\eta_{k,i}-1)/\eta_{k,i+1}$ and less than $\eta_{k,i}/(\eta_{k,i+1}-1)$. This leads to the expression

$$\kappa_{k,i,u} \in \{1, \ldots, \eta_{k,i+1}-1\} \ \wedge \ \kappa_{k,i+1,u} \in \left(\frac{\eta_{k,i}-1}{\eta_{k,i+1}}\kappa_{k,i,u}, \frac{\eta_{k,i}}{\eta_{k,i+1}-1}\kappa_{k,i,u}\right) \cap \{1, \ldots, \eta_{k,i}-1\}, \tag{29}$$

which describes how to obtain $\kappa_{k,i,u}$, $\kappa_{k,i+1,u}$, and eventually $\mathbf{K}_{k,i}$ in an algorithmic way. The algorithm has to compute all candidate pairs $(\kappa_{k,i+1,u}, \kappa_{k,i,u})$ defined by (29), remove duplicates which lead to the same rational numbers and sort the remaining unique values. On the basis of the elements of $\mathbf{K}_{k,i}$, the intervals of the ratios of the sublot sizes can be derived in which the transport paths remain fixed:

$$\left[0, \frac{\kappa_1}{\hat{\kappa}_1}\right], \left(\frac{\kappa_1}{\hat{\kappa}_1}, \frac{\kappa_2}{\hat{\kappa}_2}\right], \ldots, \left(\frac{\kappa_u}{\hat{\kappa}_u}, \frac{\kappa_{u+1}}{\hat{\kappa}_{u+1}}\right], \ldots, \left(\frac{\kappa_{w-1}}{\hat{\kappa}_{w-1}}, \frac{\kappa_w}{\hat{\kappa}_w}\right], \left(\frac{\kappa_w}{\hat{\kappa}_w}, +\infty\right), \tag{30}$$

where $\kappa_u \equiv \kappa_{k,i+1,u}$, $\hat{\kappa}_u \equiv \kappa_{k,i,u}$ (for shorthand), $w = |\mathbf{K}_{k,i}|$. The bounds 0 and $+\infty$ symbolize the actual absence of a bound from below and above, respectively, because the ratio $a_{k,i+1}/a_{k,i}$ is ultimately bounded by the extreme values of $a_{k,i+1}$ and $a_{k,i}$, thus, a duplication of this constraint in (30) would be redundant. The intervals are left-open and right-closed in general, which is a direct consequence of (1), because for each ratio $r = a_{k,i+1}/a_{k,i}$, there exists an $\varepsilon > 0$ such that $\mathcal{T}(k,i,r) = \mathcal{T}(k,i,r-\varepsilon)$, but this is not true in the case of the condition $\mathcal{T}(k,i,r) = \mathcal{T}(k,i,r+\varepsilon)$.

For instance, in the case shown in Figure 2, there are sublot numbers $\eta_{k,i} = 3$ and $\eta_{k,i+1} = 7$, thus $\mathbf{K}_{k,i} = \left\{\frac{1}{3}, \frac{2}{5}\right\}$ according to (29), and the sequence of the intervals of the ratio $r = a_{k,i+1}/a_{k,i}$ in which the transport path configuration remains fixed has the form: $\left[0, \frac{1}{3}\right], \left(\frac{1}{3}, \frac{2}{5}\right], \left(\frac{2}{5}, +\infty\right)$. It can be assumed that in the considered example r is exactly equal to $\frac{2}{5}$ (Fig. 2). Therefore, the ratio $a_{k,i+1}/a_{k,i}$ is tight at the right bound of the interval $\left(\frac{1}{3}, \frac{2}{5}\right]$. Indeed, the actual transport path configuration is $\{(1,1),(2,3),(3,6)\}$, but in the result of an arbitrary small increase of $a_{k,i+1}$ or an arbitrary small decrease of $a_{k,i}$, it will change to $\{(1,1),(2,3),(\mathbf{3},\mathbf{5})\}$.

The advantage of expression (29) is that it defines $\mathbf{K}_{k,i}$ as a function of only two integers $\eta_{k,i}$ and $\eta_{k,i+1}$. The same pairs of the values $\eta_{k,i}, \eta_{k,i+1}$ can repeat many times for different operations in a solution and for many consecutive solutions in an iterative optimization procedure. Because of that, it is time effective to generate the set $\mathbf{K}_{k,i}$ for a given pair $(\eta_{k,i}, \eta_{k,i+1})$ once, save it in memory and read next times.

Two other cases of possible value combinations in the pair $(\eta_{k,i}, \eta_{k,i+1})$ need to be considered. The first one is the case when at least one value from the pair is maximal, i.e., equal to $g(k,i)$ or $g(k,i+1)$, respectively, as defined in the first variant of (28), and the other value, if any, conforms the second variant. In this case, the interval of variability of the ratio $a_{k,i+1}/a_{k,i}$ will be additionally bounded in comparison with the one used in expression (28), because $d_{k,i} \geq e_k/\eta_{k,i}$. Thus, a different expression should be used to define the set $\mathbf{K}^*_{k,i}$ appropriate for this case, instead of $\mathbf{K}_{k,i}$ defined by (29). However, for any pair $(\eta_{k,i}, \eta_{k,i+1})$, the relation $\mathbf{K}^*_{k,i} \subseteq \mathbf{K}_{k,i}$ holds. As the result, the set $\mathbf{K}_{k,i}$ can be used in this case as well, instead of $\mathbf{K}^*_{k,i}$, because it includes the correct elements $\mathbf{K}^*_{k,i} \cap \mathbf{K}_{k,i}$, whereas the redundant elements from $\mathbf{K}_{k,i} \setminus \mathbf{K}^*_{k,i}$ lead to infeasible LP instances in which the sublot size ratio is incoherent with the bounds of the sublot sizes. These instances are dropped in the second phase of the algorithm. Using a dedicated expression for $\mathbf{K}^*_{k,i}$ may perhaps result in a more efficient algorithm, however, the simpler solution has been exploited in the presented implementation, and the verification of the second option remains a subject of future work. In the last possible case, at least one operation in the pair $(O_{k,i}, O_{k,i+1})$ is not split, i.e., it consists of a single sublot, as given in the third variant of (28). In this case, only one fixed transport path configuration between $O_{k,i}$ and $O_{k,i+1}$ is possible for all allowable values of the ratio $a_{k,i+1}/a_{k,i}$, thus formally

$$(\eta_{k,i} = 1 \vee \eta_{k,i+1} = 1) \quad \Longrightarrow \quad \mathbf{K}_{k,i} = \emptyset\,. \tag{31}$$

Finally, the discrete search space can be defined as a triple $\mathcal{D} = (\Upsilon, \Theta, \Lambda)$, where

$$\Upsilon = \{\eta_{k,i} : \ k \in \{1,\ldots,n\}\,,\ i \in \{1,\ldots,o(k)\}\,,\ \eta_{k,i} \in \{1,\ldots,g(k,i)\}\}$$

is the set consisting of the sublot numbers of all operations,

$$\Theta = \{\mathbf{K}_{k,i} : \ k \in \{1,\ldots,n\}\,,\ i \in \{1,\ldots,o(k)-1\}\}$$

is the family of sets $\mathbf{K}_{k,i}$ defined on the basis of $\eta_{k,i} \in \Upsilon$ according to (29) and (31), and

$$\Lambda = \{\lambda_{k,i} : \ k \in \{1,\ldots,n\}\,,\ i \in \{1,\ldots,o(k)-1\}\,,\ \lambda_{k,i} \in \{1,\ldots,|\mathbf{K}_{k,i}|+1\}\}$$

is the set of transport path *configuration indicators* for each pair of operations $O_{k,i}, O_{k,i+1} \in \mathbf{O}$.

The sequence (30) reveals that there are $|\mathbf{K}_{k,i}| + 1$ possible transport path configurations and related intervals of the ratio $a_{k,i+1}/a_{k,i}$. In particular, it is true as well, if $\mathbf{K}_{k,i} = \emptyset$, then $\lambda_{k,i}$ has to be equal to 1 which represents a single possible configuration. If $\mathbf{K}_{k,i} \neq \emptyset$, the value of the configuration indicator $\lambda_{k,i}$ is the interval number in the sequence (30). Therefore, a fixed position $\rho \in \mathcal{D}$ imposes bounds on the ratio $a_{k,i+1}/a_{k,i}$ as follows:

$$\mathbf{K}_{k,i} \neq \emptyset \ \wedge \ \lambda_{k,i} \geq 2 \qquad \Longrightarrow \qquad \kappa_{k,i,\lambda_{k,i}-1}\, a_{k,i+1} \ > \ \kappa_{k,i+1,\lambda_{k,i}-1}\, a_{k,i} \,, \tag{32}$$

$$\mathbf{K}_{k,i} \neq \emptyset \ \wedge \ \lambda_{k,i} \leq |\mathbf{K}_{k,i}| \qquad \Longrightarrow \qquad \kappa_{k,i,\lambda_{k,i}}\, a_{k,i+1} \ \leq \ \kappa_{k,i+1,\lambda_{k,i}}\, a_{k,i} \,, \tag{33}$$

where $(k,i) :\ k \in \{1,\ldots,n\}\,,\ i \in \{1,\ldots,o(k)-1\}$.

The position ρ unambiguously determines the transport path configuration as well, which is described by the function

$$\mathcal{T}_{\mathcal{D}}(k,i) = \begin{cases} \mathcal{T}(k,i,\frac{\kappa_{k,i+1,\lambda_{k,i}}}{\kappa_{k,i,\lambda_{k,i}}})\,, & \text{for} \quad \mathbf{K}_{k,i} \neq \emptyset \wedge \lambda_{k,i} \leq |\mathbf{K}_{k,i}|\,, \\ \mathcal{T}(k,i,\frac{e_k}{d_{k,i}})\,, & \text{for} \quad \mathbf{K}_{k,i} \neq \emptyset \wedge \lambda_{k,i} = |\mathbf{K}_{k,i}| + 1\,, \\ \{(\eta_{k,i}, 1)\}\,, & \text{for} \quad \mathbf{K}_{k,i} = \emptyset\,, \end{cases}$$

For each interval from the sequence (30), except for the last one, the configuration can be easily derived by substituting the ratio $r = a_{k,i+1}/a_{k,i}$ in (1) with the proportion $\kappa_{k,i+1,\lambda_{k,i}}/\kappa_{k,i,\lambda_{k,i}}$ from the right bound of the interval determined by $\lambda_{k,i}$. The last interval is neither left- nor right-closed, but it includes the largest possible value of $r = e_k/d_{k,i}$ which can be used in the same way. If $\mathbf{K}_{k,i} = \emptyset$, i.e., at least one of the consecutive operations $O_{k,i}, O_{k,i+1} \in \mathbf{O}$ is not split, then only one configuration is possible and it consists of a single transport path connecting the last sublot of $O_{k,i}$ with the first sublot of $O_{k,i+1}$.

A distinct LP subproblem is generated by each $\rho \in \mathcal{D}$. The formulation involves two types of decision variables

$$\alpha_{k,i} \in \mathbf{a}(\eta_{k,i})\,, \gamma_{k,i,b} \in [0, \mathcal{M}], \quad (k,i,b) :\ k \in \{1,\ldots,n\}\,,\ i \in \{1,\ldots,o(k)\}\,,\ b \in \{1,\ldots,\eta_{k,i}\}$$

which represent the sublot sizes assigned to the operations $O_{k,i}$ and the start times of the sublots $O_{k,i,b}$, respectively, similarly as in the case of the MILP model. The objective

$$\sum_{O_{k,i} \in \mathbf{O}} \alpha_{k,i} \longrightarrow \max!$$

is the same and the constraints are as follows:

$$\begin{aligned} &\gamma_{k,i+1,v} \geq \gamma_{k,i,u} && (k,i,u,v,p_1,p_2) :\ k \in \{1,\ldots,n\}\,,\ i \in \{1,\ldots,o(k)-1\}\,, \\ &\quad + \alpha_{k,i}^{(u)}\, p_{k,i,p_1} + t_{p_1,p_2}\,, && \pi_{p_1,q_1} = O_{k,i} \in \mathbf{O}_k,\ \pi_{p_2,q_2} = O_{k,i+1} \in \mathbf{O}_k\,, (u,v) \in \mathcal{T}_{\mathcal{D}}(k,i)\,, \end{aligned} \tag{34}$$

$$\begin{aligned} &\kappa_{k,i,\lambda_{k,i}-1}\, \alpha_{k,i+1} \geq && (k,i) :\ k \in \{1,\ldots,n\}\,, \\ &\quad \kappa_{k,i+1,\lambda_{k,i}-1}\, \alpha_{k,i}\,, && i \in \{1,\ldots,o(k)-1\}\,,\ \mathbf{K}_{k,i} \neq \emptyset,\ \lambda_{k,i} \geq 2\,, \end{aligned} \tag{35}$$

$$\begin{aligned} &\kappa_{k,i,\lambda_{k,i}}\, \alpha_{k,i+1} \leq && (k,i) :\ k \in \{1,\ldots,n\}\,, \\ &\quad \kappa_{k,i+1,\lambda_{k,i}}\, \alpha_{k,i}\,, && i \in \{1,\ldots,o(k)-1\}\,,\ \mathbf{K}_{k,i} \neq \emptyset,\ \lambda_{k,i} \leq |\mathbf{K}_{k,i}|\,, \end{aligned} \tag{36}$$

$$\begin{aligned} &\gamma_{k,i,b+1} \geq \gamma_{k,i,b} + \alpha_{k,i}^{(b)}\, p_{k,i,p}\,, && (k,i,p,b) :\ p \in \{1,\ldots,r\}\,,\ q \in \{1,\ldots,z(p)\}\,, \\ & && O_{k,i} = \pi_{p,q} \in \mathbf{O},\ b \in \{1,\ldots,\eta_{k,i}-1\}\ , \end{aligned} \tag{37}$$

$$\begin{aligned}\gamma_{k_2,i_2,1} \geq \alpha^{(b)}_{k_1,i_1}\, p_{k_1,i_1,p} \qquad & (k_1,i_1,k_2,i_2,p) :\ p \in \{1,\dots,r\}\,,\ q \in \{1,\dots,z(p)-1\}\,, \\ + s_{p,k_1,i_1,k_2,i_2} + \gamma_{k_1,i_1,b}\,, \qquad & O_{k_1,i_1} = \pi_{p,q} \in \mathbf{O},\ O_{k_2,i_2} = \pi_{p,q+1} \in \mathbf{O},\ b = \eta_{k_1,i_1}\,,\end{aligned} \tag{38}$$

$$\gamma_{k,i,b} + \alpha^{(b)}_{k,i}\, p_{k,i,p} \leq \mathcal{M}\,, \qquad (k,i,p) :\ p \in \{1,\dots,r\}\,,\ q = z(p),\ O_{k,i} = \pi_{p,q},\ b = \eta_{k,i}\,, \tag{39}$$

where the expression

$$\alpha^{(b)}_{k,i} = \begin{cases} \alpha_{k,i}\,, & \text{for} \quad b < \eta_{k,i}\,, \\ e_k - (\eta_{k,i} - 1)\,\alpha_{k,i}\,, & \text{for} \quad b = \eta_{k,i} \end{cases}$$

represents the actual size of an individual sublot, and it is used for conciseness.

The constraints (34) model the transport paths imposing that the start times of the sublots $O_{k,i+1,v}$ have to be not less than the start times of the previous sublots $O_{k,i,u}$ increased by the related processing and transport delays. In this model, the configuration of transport paths is fixed in advance and given as a set of pairs (u,v) by the function $\mathcal{T}_{\mathcal{D}}(k,i)$. The ratio $a_{k,i+1}/a_{k,i}$ of sublot sizes of consecutive operations is lower and upper bounded by (35) and (36), respectively, according to (32) and (33). The constraints (37), (38), and (39) ensure correct start-to-start time relationships between the sublots processed on the same machine and impose the required makespan value in exactly the same way as (25), (26), and (27) in the MILP model, respectively.

The described LP model has to be used in combination with a master discrete optimization procedure which explores the points of the search space $\mathcal{D}$. A tabu search metaheuristic has been applied for this purpose. Its implementation follows once more the general structure represented by Pseudocode 1, except that the value of the objective function is maximized. For each position in the space $\mathcal{D}$, selected by the tabu search algorithm, the underlying LP problem is solved and the result is considered as the value of the objective function for this point. The time consuming tasks of constructing and solving the LP model are executed in parallel using a pool of computational threads. The other details of the implementation are described below.

Four elementary moves have been defined to describe the position changes in the space $\mathcal{D}$:

- The move $\delta_1(k,i)$ decreases the number of sublots of the operation $O_{k,i}$ from $\eta_{k,i}$ to $\eta^*_{k,i} \geq 1$, where $j = \eta_{k,i} - \eta^*_{k,i}$ is picked randomly. In the result of this modification, the sets $\mathbf{K}_{k,i}$ (for $i > 1$) and $\mathbf{K}_{k,i+1}$ (for $i < o(k)$), as well as the ranges of the values of the related configuration indicators $\lambda_{k,i}$ and $\lambda_{k,i+1}$ also change, hence, new values are randomly chosen for these parameters likewise.

- The move $\delta_2(k,i)$ is analogous to $\delta_1(k,i)$, except that the value of $\eta_{k,i}$ is increased.

- The move $\delta_3(k,i)$ decreases the configuration indicator $\lambda_{k,i}$ by a randomly chosen integer. This move has no impact on $\eta_{k,i}$ and $\mathbf{K}_{k,i}$.

- The move $\delta_4(k,i)$ is analogous to $\delta_3(k,i)$, but the value of $\lambda_{k,i}$ is increased.

The moves $\delta_1(k,i)$ and $\delta_2(k,i)$ are defined for $k \in \{1,\dots,n\}\,, i \in \{1,\dots,o(k)\}$, whereas $\delta_3(k,i)$ and $\delta_4(k,i)$ for $k \in \{1,\dots,n\}\,, i \in \{1,\dots,o(k)-1\}$. The described moves have no effect if the decreased or increased parameter has already the minimum or maximum value, respectively. The optimization algorithm starts with the neighborhood

$$\mathbf{N}^{\mathrm{A}} = \bigcup_{k=1}^{n} \bigcup_{i=1}^{o(k)} \{\delta_1(k,i)\}\,,$$

which consists only of moves $\delta_1(k,i)$. In this way, the value of the objective function is increased rapidly at the beginning of the optimization, because the sublots have initially minimum sizes and increasing these sizes by decreasing the number of sublots is an advantageous direction of exploring the search space. This neighborhood is exploited as long as each subsequent solution is better than the previous one. After this hill climbing phase, the algorithm switches to the full neighborhood

$$\mathbf{N}^{\mathrm{B}} = \bigcup_{k=1}^{n} \bigcup_{i=1}^{o(k)} \{\delta_1(k,i)\} \cup \bigcup_{k=1}^{n} \bigcup_{i=1}^{o(k)} \{\delta_2(k,i)\} \cup \bigcup_{k=1}^{n} \bigcup_{i=1}^{o(k)-1} \{\delta_3(k,i)\} \cup \bigcup_{k=1}^{n} \bigcup_{i=1}^{o(k)-1} \{\delta_4(k,i)\} .$$

It has been experimentally checked that the neighborhood based on random changes of the parameter values is more effective in terms of the optimization speed than changing the values by one or examining all possible values in the neighborhood simultaneously.

The tabu attribute has been defined as a 6-tuple consisting of the main parameters of a move in the space $\mathcal{D}$. Before the solution is modified by a move related to an operation $O_{k,i}$, the value $(k, i, \eta_{k,i}, \eta_{k,i-1}, \eta_{k,i+1}, \lambda_{k,i})$ is added to the tabu list, under the assumption that $\eta_{k,0} = \eta_{k,o(k)+1} = \lambda_{k,o(k)} = 0$. A new move is forbidden if it would restore any of the 6-tuple parameter configurations currently in the tabu list. Because the sizes of the neighborhoods $\mathbf{N}^{\mathrm{A}}$ and $\mathbf{N}^{\mathrm{B}}$ are of order $O(|\mathbf{O}|)$, the tabu list length is fixed at the value $\nu|\mathbf{O}|$. Experiments have shown that the algorithm works fine for $\nu = 1$.

The stopping condition has been defined in exactly the same way as for the scheduling tabu search implementation (13).

6.2 Maximization of the number of non-split lots

A basic model for the optimization of the objective $f^{\mathrm{B}}_{\mathrm{size}}$ has been derived from the previously developed MILP model (6.1.1). To that end, the following changes have been introduced to the model:

- binary decision variables

$$\beta_{k,i} \in \{0, 1\}, \quad (k, i) : \; k \in \{1, \ldots, n\}, \; i \in \{1, \ldots, o(k) - 1\}, \tag{40}$$

  have been added such that, if $\beta_{k,i} = 1$, then $O_{k,i} \in \mathbf{O}$ is not split, i.e., $\alpha_{k,i} = e_k$,

- the mentioned relationship between $\beta_{k,i}$ and $\alpha_{k,i}$ has been imposed by the additional constraint

$$\alpha_{k,i} \geq e_k \, \beta_{k,i}, \quad (k, i) : \; k \in \{1, \ldots, n\}, \; i \in \{1, \ldots, o(k) - 1\}, \tag{41}$$

- the objective has been modified to the form

$$\sum_{O_{k,i} \in \mathbf{O}} \beta_{k,i} \longrightarrow \max! \tag{42}$$

This optimization approach, in which continuous sublot sizes are permitted, will be denoted as MILP-CN in the remainder of the text.

The MILP-CN model appeared to be too difficult to be effectively solved in the case of larger problem instances. Hence, a simplified model has been developed (MILP-MM), in which the sublot size can be only either minimal (i.e., equal to $d_{k,i}$) or maximal (i.e., equal to e_k). This restriction reduces the solution space and the best results, potentially available by the approach MILP-CN, may be missed. However, because of efficiency issues, the MILP-MM approach may be better in practice. The used simplification is based on the heuristic assumption that, if a sublot size cannot be maximal due to model constraints, it should be minimal to not impede maximization of other sublots. The MILP-MM model has been obtained by a modification of the original MILP formulation (6.1.1) as well. The variables $\beta_{k,i}$ (40) have been introduced again and the objective has been reformulated according to expression (42). Instead of introducing the constraints (41), however, the variables $\alpha_{k,i}$ have been removed from the model and replaced with the linear expressions

$$d_{k,i} + (e_k - d_{k,i})\beta_{k,i}, \quad (k,i) : \ k \in \{1,\ldots,n\}\,,\ i \in \{1,\ldots,o(k)-1\}\,,$$

so that a sublot size has to be minimal if the respective $\beta_{k,i} = 0$ and maximal otherwise.

As in the case of the previous objective functions f_{sched} and $f^{\text{A}}_{\text{size}}$, an alternative algorithm has also been developed for the function $f^{\text{B}}_{\text{size}}$ which does not depend on third-party proprietary software. In this implementation, it has been assumed again that the sublot sizes can be only minimal or maximal. The algorithm generates solutions in a greedy fashion (Pseudocode 2). The generation starts from the solution with minimal sizes of all sublots. Then a random se-

Pseudocode 2: Greedy generation of solutions for the objective function $f^{\text{B}}_{\text{size}}$

```
orderedOperations := getListOfAllOperations();
permutedOperations := permuteRandomly(orderedOperations);
foreach $O_{k,i}$ in orderedOperations do
    sublotSizes[$O_{k,i}$] := $d_{k,i}$;
end
objective := 0;
foreach $O_{k,i}$ in permutedOperations do
    sublotSizes[$O_{k,i}$] := $e_{k,i}$;
    makespan := calculateMakespan(sublotSizes);
    if makespan ≤ $\mathcal{M}$ then
        objective := objective + 1;
    else
        sublotSizes[$O_{k,i}$] := $d_{k,i}$;
    end
end
return {sublotSizes, objective};
```

quence of operations is picked, and the algorithm tries to change the sublot sizes from minimal to maximal, going in the order of this sequence. The maximal size is set if it does not increase the makespan value over the result obtained in the scheduling stage ($\mathcal{M}$), and the minimal size is left otherwise. The makespan value is calculated using the graph model $\mathbf{G}(\Gamma, \Pi)$ (Section 4). The graph structure has to be reconstructed for each calculation, because the number of vertices and the configuration of arcs changes as the result of the modification of a sublot size. This reconstruction, together with the calculation of the length of the longest path (i.e., the makespan) takes $O(|\mathbf{O}|)$ time. The algorithm calculates the makespan value once after the change of the sublot size of each operation, thus, its time complexity is equal to $O(|\mathbf{O}|^2)$. The structure of the algorithm asserts that any obtained solution is feasible. The final implementation of the algorithm uses a parallel computation to exploit the multicore CPUs. The algorithm can be augmented by a master procedure for searching within the permutation space, but it was not needed for the considered benchmark instances, because the optimal values of the objective function repeated frequently in the sets of solutions obtained using a random generation of permutations.

Table 1: Parameters of the benchmark instances

			$\sum a_{k,i}$		$\sum \lceil \frac{e_k}{a_{k,i}} \rceil$		$d_{k,i}$		$g(k,i)$		$o(k)$		$\lvert\mathbf{M}_{k,i}\rvert$	
Inst.	n	r	LB	UB	LB	UB	min	max	min	max	min	max	min	max
1	5	5	2026	3563	35	79	40	80	1	3	6	8	2	4
2	5	5	1003	3445	33	135	20	40	2	6	6	8	2	4
3	5	5	516	3756	36	293	10	20	4	12	6	8	2	4
4	5	5	251	3602	34	532	5	10	8	24	6	8	2	4
5	10	10	6302	9864	101	218	40	80	1	3	6	8	2	4
6	10	10	3157	10858	105	428	20	40	2	6	8	12	4	6
7	10	10	1437	9400	96	705	10	20	4	12	8	12	4	6
8	10	10	702	9833	98	1484	5	10	8	24	8	12	4	6
9	10	15	12830	19751	205	441	40	80	1	3	8	12	4	6
10	10	15	5790	19177	190	742	20	40	2	6	8	12	4	6
11	10	15	3205	20939	211	1555	10	20	4	12	16	24	4	6
12	10	15	1594	21021	214	3061	5	10	8	24	16	24	4	6
13	15	15	17744	32024	304	722	40	80	1	3	16	24	4	6
14	15	15	9552	34233	321	1348	20	40	2	6	16	24	4	6
15	15	15	4535	30484	299	2251	10	20	4	12	16	24	4	6
16	15	15	2248	29414	298	4275	5	10	8	24	16	24	4	6
17	20	20	23587	40179	394	895	40	80	1	3	16	24	4	6
18	20	20	11971	39370	404	1590	20	40	2	6	16	24	4	6
19	20	20	5946	40912	395	3021	10	20	4	12	16	24	4	6
20	20	20	3002	40390	409	5993	5	10	8	24	16	24	4	6

Indices (k, i) : $k \in \{1, \ldots, n\}$, $i \in \{1, \ldots, o(k)\}$,

LB / UB (lower / upper bound) – extreme values reachable in the solutions,

min / max – statistical parameters over the sets of jobs or operations

7 Computational experiments

7.1 Benchmark instances

A set of 20 benchmark instances has been generated for the experiments (Table 1). The parameters of the instances conform a regular pattern: there are 5 groups with different numbers of jobs and machines (5x5, 10x10, 10x15, 15x15, 20x20), and there are 4 instances in each group with different intervals of possible sublot sizes. Except for the numbers of jobs and machines which are explicitly fixed, the individual values of all other parameters have been randomly selected from the given intervals. The intervals of the values $d_{k,i}$, $o(k)$, and $|\mathbf{M}_k|$ are instance-dependent (Table 1), whereas the other intervals are common for all instances: $e_k \in [80, 120]$, $t_{p_1,p_2} \in [5, 30]$, $s_{p,k_1,i_1,k_2,i_2} \in [20, 40]$, $p_{k,i,p} \in [15, 50]$. The parameters have been chosen as integers using a uniform discrete distribution. According to the problem formulation, real values are allowed for some parameters, but it is not important in the context of the performed experiments and integers are more convenient as the values stored in the text files of the instances. The maximum number of operation sublots $g(k, i)$ varies from $[1, 3]$ to $[8, 24]$, as the result of the respective values e_k and $d_{k,i}$. On the basis of the instance parameters, bounds can be derived on values important for the optimization problem. The bounds on the value $\sum a_{k,i}$ also concern the objective function $f^{\text{A}}_{\text{size}}$, and in addition a lower bound on $\sum \lceil \frac{e_k}{a_{k,i}} \rceil$, equal to $|\mathbf{O}|$, represents the maximum possible value of $f^{\text{B}}_{\text{size}}$.

7.2 Organization of experiments

All implemented scheduling and lot sizing approaches are collected in Table 2, together with some basic information about the related computational experiments.

The experiments performed with the use of CPO and CPLEX solvers have been executed

Table 2: Overview of implementations and experiments

Problem	Objective	Options	Implementation	Model	Solver	t^{TOTAL}	t^{RUN}	Execution
SCHEDULING	f_{sched}	no lot streaming	**CPO-NLS**	OCSP	CPO	≤ 5 h	–	single
			TS-NLS	graph	–	$= 5$ h	≥ 5 m	repeated
		lot streaming	**CPO-LS**	OSCP	CPO	≤ 5 h	–	single
			TS-LS	graph	–	$= 5$ h	≥ 5 m	repeated
LOT SIZING	$f^{\mathrm{A}}_{\mathrm{size}}$	continuous lot sizes	**MILP**	MILP	CPLEX	≤ 5 h	–	single
			TS+LP	LP	CLP	$= 5$ h	≥ 5 m	repeated
	$f^{\mathrm{B}}_{\mathrm{size}}$	cnt. lot sizes	**MILP-CN**	MILP	CPLEX	≤ 1 h	–	single
		min/max lot sizes	**MILP-MM**	MILP	CPLEX	≤ 1 h	–	single
			GR	graph	–	$= 1$ h	–	repeated

t^{TOTAL} – total time of computation related to a benchmark instance,
t^{RUN} – time of execution of a single optimization task,
OCSP – optimal constraint satisfaction problem, (MI)LP – (mixed-integer) linear programming problem,
CPO – IBM ILOG CP Optimizer solver, CPLEX – IBM ILOG CPLEX solver,
CLP – COIN-OR linear programming solver

once for each benchmark instance, because the solvers used deterministic strategies and repetitions of the computation would result in the same solutions. A time limit of 5 hours was set, except for MILP-CN and MILP-MM, i.e., the optimization of each benchmark instance was interrupted after 5 hours and the best found solution was registered if the solver had not found an optimal solution earlier. For the implementations MILP-CN and MILP-MM, the time limit was set to 1 hour, because the problems with the objective function $f^{\mathrm{B}}_{\mathrm{size}}$ are solved faster.

The experiments based on the tabu search approach, i.e., the scheduling approach without lot streaming (TS-NLS), and with lot streaming (TS-LS), as well as lot sizing with the objective function $f^{\mathrm{A}}_{\mathrm{size}}$ (TS+LP) have been performed several times (*repeated*, see Table 2) up to a total execution time equal to 5 hours. The repetition is justified, as the results may differ, because of, among others, randomly generated initial solutions. The values $t^{\mathrm{MIN}} = 5$ min, $stag = 1$ were set for both the algorithms (Subsections 5.2, 6.1.2), hence each repetition of the algorithms lasted at least 5 minutes. Additionally, the control procedure of the tabu list length for the scheduling algorithm (Subsection 5.2) has been configured with the following parameter values: $tl^{\mathrm{I}} = 10$, $tl^{\mathrm{R}} = 5$. The algorithm GR was executed for 1 hour per instance, similarly as in the case of the approaches MILP-CN and MILP-MM.

All the computations have been performed on the same PC with the processor Intel Core i7-4710MQ and 16 GB of RAM memory. All the directly implemented algorithms (TS, TS+LP, GR) used the full computational power of the processor during the whole execution due to their concurrent implementation. The computations performed by the IBM ILOG solvers (CPO, MILP, MILP-CN, MILP-MM) used, as a rule, the full computational power as well. During the experiments, the CPU had no other significant load.

7.3 Results

The results of scheduling are summarized in Table 3. For all combinations of the approaches CPO/TS and the scheduling modes NLS/LS, the values of the objective function f_{sched} are given. There are also presented other data related to the objective: the time at which CPO

Table 3: Scheduling without and with lot streaming – CPO vs TS

	CPO					TS									CPO vs TS	
	NLS		LS			NLS – f_{sched}				LS – f_{sched}					NLS	LS
Inst.	f_{sched}	t^{LI}	f_{sched}	t^{LI}	Δ_1	min	max	$\frac{\sigma}{\mu}$	R	min	max	$\frac{\sigma}{\mu}$	R	Δ_2	Δ_3	Δ_4
		[min]		[min]	[%]			[%]				[%]		[%]	[%]	[%]
1	20411	0	19385	0	5.0	20411	20835	0.3	60	19385	19385	0.0	60	5.0	0.0	0.0
2	23245	0	20343	28	12.5	23245	23602	0.4	60	20343	21174	0.9	60	12.5	0.0	0.0
3	25348	0	21136	37	16.6	25348	25627	0.3	60	20573	21801	1.3	60	18.8	0.0	2.7
4	23585	0	19278	250	18.3	23585	24236	0.4	60	19088	20416	1.0	59	19.1	0.0	1.0
5	26161	261	24832	232	5.1	26344	26704	0.3	37	24487	24849	0.3	41	7.0	−0.7	1.4
6	30052	15	26021	260	13.4	30052	30106	0.0	52	24875	25611	0.5	48	17.2	0.0	4.4
7	24246	66	21104	116	13.0	24454	25075	0.5	39	20802	21311	0.6	39	14.9	−0.9	1.4
8	28223	9	23470	260	16.8	28256	28423	0.1	57	22513	23003	0.6	34	20.3	−0.1	4.1
9	59806	3	46528	225	22.2	59806	59806	0.0	60	46528	46528	0.0	60	22.2	0.0	0.0
10	52708	17	36693	120	30.4	52708	52708	0.0	60	33587	34430	0.5	29	36.3	0.0	8.5
11	53064	5	36589	261	31.0	52841	52841	0.0	60	34304	34945	0.5	35	35.1	0.4	6.2
12	53421	282	38166	284	28.6	53356	53401	0.0	54	33293	33896	0.6	19	37.6	0.1	12.8
13	60427	247	55278	275	8.5	60207	60825	0.3	50	52192	53560	0.8	18	13.3	0.4	5.6
14	65495	96	60048	296	8.3	62357	62541	0.2	2	54087	55270	0.8	6	13.3	4.8	9.9
15	55142	255	55086	296	0.1	54263	55053	0.4	27	46606	48130	1.2	6	14.1	1.6	15.4
16	56418	134	56730	297	−0.6	55310	55956	0.4	28	44650	45113	0.4	6	19.3	2.0	21.3
17	56894	250	52969	276	6.9	54533	55453	0.6	8	49932	51366	0.8	18	8.4	4.1	5.7
18	60036	245	54483	213	9.2	58092	58899	0.4	24	47966	49871	1.3	7	17.4	3.2	12.0
19	62939	277	57527	285	8.6	61631	62525	0.3	41	47325	47891	0.6	3	23.2	2.1	17.7
20	60719	268	63275	300	−4.2	59307	60674	0.9	23	47947	47947	0.0	1	19.2	2.3	24.2

$\Delta_1 = \frac{f_{\text{sched}}^{\text{NLS,CPO}} - f_{\text{sched}}^{\text{LS,CPO}}}{f_{\text{sched}}^{\text{NLS,CPO}}}$, $\Delta_2 = \frac{f_{\text{sched}}^{\text{NLS,TS}} - f_{\text{sched}}^{\text{LS,TS}}}{f_{\text{sched}}^{\text{NLS,TS}}}$, $\Delta_3 = \frac{f_{\text{sched}}^{\text{NLS,CPO}} - f_{\text{sched}}^{\text{NLS,TS}}}{f_{\text{sched}}^{\text{NLS,CPO}}}$, $\Delta_4 = \frac{f_{\text{sched}}^{\text{LS,CPO}} - f_{\text{sched}}^{\text{LS,TS}}}{f_{\text{sched}}^{\text{LS,CPO}}}$,

LS / NLS – no lot streaming / lot streaming, R – number of repetitions of the algorithm,

in this and consecutive tables: $\frac{\sigma}{\mu}$ – coefficient of variation, t^{LI} – time of the last improvement of the respective objective

generated the last solution, as well as statistical parameters concerning the sets of solutions obtained by TS, i.e., the minimum (the best value) and maximum, the coefficient of variation, and the number of repetitions of the algorithm. Ratios are also given which compare the results: NLS vs LS for CPO (Δ_1), NLS vs LS for TS (Δ_2), CPO vs TS for NLS (Δ_3), and CPO vs TS for LS (Δ_4). The times of the last improvement (CPO) and the numbers of repetitions (TS) demonstrate that the smallest instances 1-5 were processed very quickly by the algorithms, especially in the case of NLS+CPO, whereas the larger instances required significantly more amount of time to be solved. The best results obtained by CPO and TS are identical or very similar across the instances 1-13 NLS and 1, 2, 9 LS (Δ_3). In general, TS provided better or much better results, particularly in the case of LS, for which the difference exceeds 10% (instances 12, 15, 16, 18-20). Hence, the results of LS+CPO for the larger instances have to be far away from the global optimum and the comparison based on the ratio (Δ_1) is not very relevant. In contrast, the value of Δ_2 is credible and represents the most important information in Table 3. It indicates the advantage of lot streaming by means of the relative makespan decrease which varies in the range from 5% to almost 40%. As one could expect, Δ_2 typically grows with the number of sublots per operation. The coefficient of variation of the best values of $f_{\text{size}}^{\text{A}}$ obtained by TS is most often less than 1% and thus, the values are well concentrated.

The outcome of the second stage of the optimization with the objective function $f_{\text{size}}^{\text{A}}$ (the sum of the sublot sizes) is presented in Table 4. For the solutions obtained using MILP, the objective function value, the upper bound, the gap, and the time of the last improvement are given. In the case of TS+LP, the statistical data related to the objective function value and the parameter t^{LI} are collected. The comparison of the used approaches is based on the ratio Δ_5 and the parameters R^{L}, R^{E}, R^{G} that represent the number of TS+LP repetitions for which the value of $f^{\mathrm{A}}_{\mathrm{size}}$ is smaller than, equal to, or greater (better) than the final result of MILP, respectively. Some statistics of the time t^{G} are also provided, after this time TS+LP solutions have a better objective function value than the final objective function value obtained by MILP for the same instance. The MILP approach provided 8 optimal solutions, some of them have been found very quickly (instances 1, 2, 5, 9) and the other ones in time a few orders greater (instances 3, 6, 13, 17). On the other hand, the results of MILP are very poor in the case of the largest instances, especially the instances with the greatest number of sublots per operation (12, 16, 20). It seems to be a consequence of the high computational complexity of the model/algorithm, because the computation time grows rapidly with the instance size. However, it is difficult to characterize the reason precisely, as the solver is proprietary software. The results of TS+LP are generally better in terms of the objective function value, i.e., this value is insignificantly less than the optimum found by MILP for some instances (6, 13, 17) and equal to or greater than the objective function value of the MILP solutions in other cases, in particular, enormously greater for instances 16 and 20. The parameters t^{LI} and t^{G} indicate that the solutions of TS+LP are generated faster. Thus, in the overall assessment, the main advantage of TS+LP is a good scalability for a broad range of instance sizes. The advantage of MILP, probably less important in practice, is the possibility of providing an optimality proof. The coefficients of variation show that the values of $f^{\mathrm{A}}_{\mathrm{size}}$ obtained by TS+LP are very well concentrated, whereas the values of the time parameters (t^{LI}, t^{G}) are highly dispersed. The ratio $\Delta^{\mathbf{A}}_{\mathbf{B}}$ compares the best value of $f^{\mathrm{A}}_{\mathrm{size}}$ from Table 4 to the value of the same objective function calculated for the solutions obtained using MILP-MM. The comparison indicates that the maximization of the function $f^{\mathrm{B}}_{\mathrm{size}}$ does not find good results for the objective function $f^{\mathrm{A}}_{\mathrm{size}}$.

The results obtained for the objective function $f^{\mathrm{B}}_{\mathrm{size}}$ are shown in Table 5. Most of the parameters are defined in the same way as in the previous table (val, UB, gap, t^{LI}). The parameter ρ is the ratio $f^{\mathrm{B}}_{\mathrm{size}}$ to $|\mathbf{O}|$ which expresses what part of all operations is not split and facilitates a comparison of the results from different instances. The outcome of MILP-CN is similar to the one obtained before for MILP with the objective function $f^{\mathrm{A}}_{\mathrm{size}}$, i.e., the smallest instances (1-6) are optimized very quickly and an optimality proof is given, whereas the results are very weak for the larger instances, especially 12, 16, and 20 – it is evident from the comparison of the MILP-CN and MILP-MM results, as the formulation of the models impose that $f^{\mathrm{B,MILP-CN}}_{\mathrm{size}} \geq f^{\mathrm{B,MILP-MM}}_{\mathrm{size}}$ for optimal solutions, which is fractured for larger instances (11, 12, 15, 16, 19, 20). The solver optimized effectively all instances using the MILP-MM model. The optimality was proved for all these cases, so the values of UB and gap are omitted in Table 5. The instances not solved effectively by MILP-CN were processed much longer by MILP-MM as well, so this confirms their difficulty. The best results of the GR approach are identical to those provided by MILP-MM and thus, they are not presented redundantly in the table. The parameter $\theta_{\min}$ specifies the percentage of occurrence of an optimal solution in the set of all solutions, and its values usually exceed 50%, except for a few examined instances (9, 13, 17). The ratio Δ_6 gives a comparison of the worst and the best objective function values. The parameter ν represents the number of solutions generated per second. Hence, the GR algorithm provides a single solution every $1/\nu$ seconds, the solution is always feasible, it is optimal with a probability estimated by $\theta_{\min}$ and, in the worst case, it has a gap from the optimal value equal to Δ_6. This algorithm is, therefore, pretty effective in comparison to MILP-MM, despite the simple implementation. The parameter $\Delta^{\mathbf{B}}_{\mathbf{A}}$ compares the best value of $f^{\mathrm{B}}_{\mathrm{size}}$ obtained by MILP-MM to the value of the same function calculated for the best solutions optimized with respect to the objective function $f^{\mathrm{A}}_{\mathrm{size}}$. In contrast to the values of $\Delta^{\mathbf{A}}_{\mathbf{B}}$ (Table 4), the ratio

Table 4: Lot sizing: sum of sublot sizes – MILP vs TS+LP

	MILP				TS+LP						MILP vs TS+LP							
	$f^{\mathrm{A}}_{\mathrm{size}}$				$f^{\mathrm{A}}_{\mathrm{size}}$			t^{LI}							t^{G}			
Inst.	val	UB	gap	t^{LI}	min	max	$\frac{\sigma}{\mu}$	min	max	$\frac{\sigma}{\mu}$	Δ_5	R^{L}	R^{E}	R^{G}	min	max	$\frac{\sigma}{\mu}$	$\Delta^{\mathbf{A}}_{\mathbf{B}}$
			[%]	[min]			[%]	[min]	[min]	[%]	[%]				[min]	[min]	[%]	[%]
1	3270	3271	0.0	0.0	3270	3270	0.0	0.0	0.2	117.3	0.0	0	60	0				10.0
2	2767	2768	0.0	0.0	2712	2767	0.6	0.0	4.6	180.4	0.0	21	36	0				22.7
3	2393	2393	0.0	4.6	2375	2393	0.3	0.1	6.5	120.2	0.0	35	20	0				36.5
4	1910	2477	29.7	299.9	1931	1942	0.1	0.2	7.6	74.7	1.7	0	0	49	0.0	0.8	91.6	31.8
5	9179	9180	0.0	0.0	9130	9179	0.1	0.1	1.1	57.6	0.0	1	59	0				12.3
6	7821	7822	0.0	51.5	7762	7816	0.2	0.3	4.0	69.4	−0.1	60	0	0				31.5
7	5765	6914	19.9	298.4	5746	5821	0.4	0.7	68.7	204.9	1.0	4	0	16	0.3	7.2	90.3	46.2
8	4838	7075	46.2	296.4	5266	5294	0.3	28.6	93.9	64.1	9.4	0	0	3	0.3	0.3	5.2	42.2
9	17470	17472	0.0	0.1	17411	17470	0.1	1.7	8.3	36.6	0.0	35	2	0				12.7
10	11724	14029	19.7	299.9	11758	11867	0.3	3.2	43.0	115.5	1.2	0	0	15	1.7	7.0	52.2	38.0
11	9541	14740	54.5	300.0	9811	9811	0.0	151.4	151.4	0.0	2.8	0	0	1	4.1	4.1	0.0	43.3
12	5589	14464	158.8	290.6	8675	8675	0.0	216.0	216.0	0.0	55.2	0	0	1	1.9	1.9	0.0	51.9
13	28158	28161	0.0	10.2	28089	28117	0.0	9.0	28.1	41.2	−0.1	9	0	0				17.2
14	23627	26742	13.2	297.9	23643	23762	0.2	14.7	52.8	44.9	0.6	0	0	5	6.2	24.5	68.0	31.7
15	17804	24861	39.6	299.8	18148	18309	0.6	18.5	137.4	107.8	2.8	0	0	2	5.6	6.1	5.9	40.8
16	2679	22223	729.6	293.8	15810	15957	0.7	48.9	126.2	62.4	495.7	0	0	2	0.3	0.4	14.9	47.9
17	35577	35581	0.0	10.7	35411	35511	0.1	5.8	33.9	53.0	−0.2	9	0	0				11.7
18	28210	33428	18.5	297.6	28272	28303	0.0	22.3	38.0	19.7	0.3	0	0	5	9.7	13.6	15.5	27.7
19	23512	35551	51.2	299.8	24418	24418	0.0	223.5	223.5	0.0	3.9	0	0	1	14.3	14.3	0.0	43.9
20	3481	31582	807.2	300.0	19056	19056	0.0	173.1	173.1	0.0	447.4	0	0	1	0.8	0.8	0.0	47.2

$\Delta_5 = \frac{f^{\mathrm{A,TS+LP(max)}}_{\mathrm{size}} - f^{\mathrm{A,MILP}}_{\mathrm{size}}}{f^{\mathrm{A,MILP}}_{\mathrm{size}}}$, $\Delta^{\mathbf{A}}_{\mathbf{B}} = \frac{f^{\mathrm{A,best}}_{\mathrm{size}} - f^{\mathrm{A,MILP-MM(B)}}_{\mathrm{size}}}{f^{\mathrm{A,best}}_{\mathrm{size}}}$, $\mathrm{gap} = \frac{\mathrm{UB} - \mathrm{val}}{\mathrm{val}}$,

val – value of the respective function, UB – upper bound,

R^{L} / R^{E} / R^{G} – number of the repetitions of the TS+LP algorithm for which the objective function values are less than / equal to / greater (better) than the objective function value obtained using MILP (realtive equality tolerance: 10^{-4}), $R^{\mathrm{L}} + R^{\mathrm{E}} + R^{\mathrm{G}}$ – total number of the repetitions of the algorithm,

t^{G} – the earliest time after which the TS+LP objective function reaches a value greater than the one obtained using MILP

$\Delta^{\mathbf{B}}_{\mathbf{A}}$ indicates that the optimization of $f^{\mathrm{A}}_{\mathrm{size}}$ usually maximizes $f^{\mathrm{B}}_{\mathrm{size}}$ as well, in some cases even better than the dedicated MILP-MM and GR approaches (instances 6, 7, 10). However, the optimization of $f^{\mathrm{A}}_{\mathrm{size}}$ is significantly more time-consuming for the larger among the examined instances and, in general, has a higher time complexity.

In Figure 4, the objective function value is presented as a function of time for selected larger instances. The plots are helpful in understanding the reason of the performance differences between the used approaches. For instances 17-20, in the case of scheduling (makespan minimization) with lot streaming, the CPO approach is remarkably slower than TS (Figure 4a) and the time limit makes it impossible to obtain a better solution using this approach. The changes of $f^{\mathrm{A}}_{\mathrm{size}}$ during the optimization based on the MILP model (Figure 4b) are much less regular, and it is difficult to predict whether a longer computation time would lead to better solutions. The objective function values change very smoothly in the case of the approaches TS and TS+LP. It is a property important in practice, because it makes it possible to detect a stagnation reliably and stop the computation when greater improvements of a solution are not expected.

In the bar plot presented in Figure 5, some important parameters of the solutions are compared. To make a comparison possible, the relative values expressed in percentages are considered. Each triple of bars represents in turn, from left to right:

- the relationship between the makespan values obtained using NLS and LS scheduling, expressed by the parameter Δ_2 (Table 3),

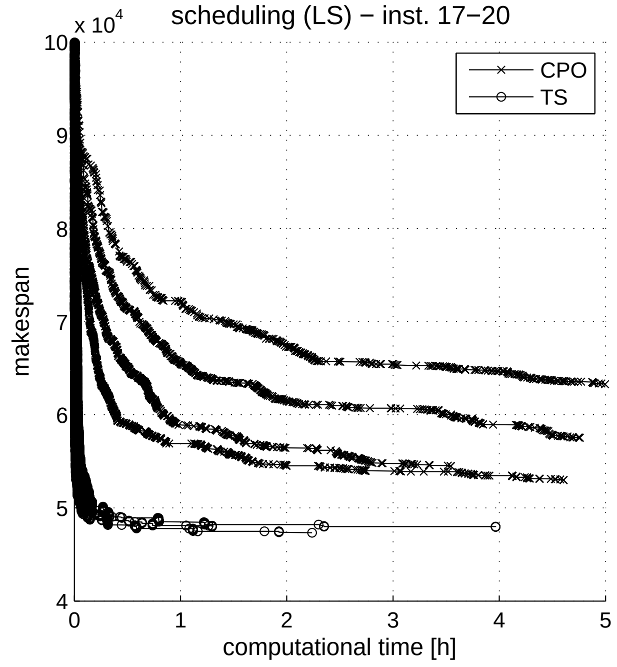
Table 5: Lot sizing: number of non-split lots – MILP-CN, MILP-MM, GR

	MILP-CN					MILP-MM			GR				$f^{\mathrm{A}}_{\mathrm{size}}$ optim	
	$f^{\mathrm{B}}_{\mathrm{size}}$					$f^{\mathrm{B}}_{\mathrm{size}}$			$f^{\mathrm{B}}_{\mathrm{size}}$					
Inst.	val	UB	gap	ρ	t^{LI}	val	ρ	t^{LI}	$\theta_{\max}$	min	Δ_6	ν	$f^{\mathrm{B}}_{\mathrm{size}}$	$\Delta^{\mathbf{B}}_{\mathbf{A}}$
			[%]	[%]	[min]		[%]	[min]	[%]		[%]	[1/s]		[%]
1	22	22	0.0	62.9	0.0	22	62.9	0.0	100.0	22	0.0	2504.7	19	13.6
2	16	16	0.0	48.5	0.0	16	48.5	0.0	66.7	15	6.3	2060.3	15	6.3
3	13	13	0.0	36.1	0.2	13	36.1	0.0	100.0	13	0.0	1174.6	13	0.0
4	11	11	0.0	32.4	0.7	11	32.4	0.1	100.0	11	0.0	783.6	11	0.0
5	72	72	0.0	71.3	0.0	62	61.4	0.0	63.3	57	8.1	465.3	55	11.3
6	39	39	0.0	37.1	7.7	31	29.5	0.1	66.7	30	3.2	282.1	34	−9.7
7	25	52	108.0	26.0	1.8	21	21.9	0.2	100.0	21	0.0	199.7	23	−9.5
8	1	56	5.5e+03	1.0	10.2	25	25.5	1.3	100.0	25	0.0	100.9	23	8.0
9	124	143	14.9	60.5	0.3	91	44.4	0.0	6.1	77	15.4	114.0	89	2.2
10	40	97	142.4	21.1	54.6	24	12.6	0.4	100.0	24	0.0	79.0	29	−20.8
11	10	118	1.1e+03	4.7	9.5	29	13.7	1.8	66.9	28	3.4	30.9	27	6.9
12	1	143	1.4e+04	0.5	43.1	29	13.6	10.2	100.0	29	0.0	15.2	23	20.7
13	167	218	30.8	54.9	58.2	132	43.4	0.1	1.4	124	6.1	43.9	125	5.3
14	99	209	111.5	30.8	58.1	89	27.7	1.1	63.8	83	6.7	22.9	89	0.0
15	35	212	504.9	11.7	57.0	74	24.7	3.0	66.8	73	1.4	14.6	72	2.7
16	1	193	1.9e+04	0.3	52.9	69	23.2	15.1	24.5	66	4.3	7.3	62	10.1
17	235	278	18.1	59.6	59.6	207	52.5	0.1	7.4	193	6.8	24.3	193	6.8
18	156	267	71.2	38.6	8.4	133	32.9	1.2	26.7	126	5.3	14.2	132	0.8
19	37	301	713.6	9.4	59.6	91	23.0	8.5	28.9	83	8.8	6.4	90	1.1
20	0	290	2.9e+14	0.0	0.1	78	19.1	29.4	47.1	71	9.0	3.3	69	11.5

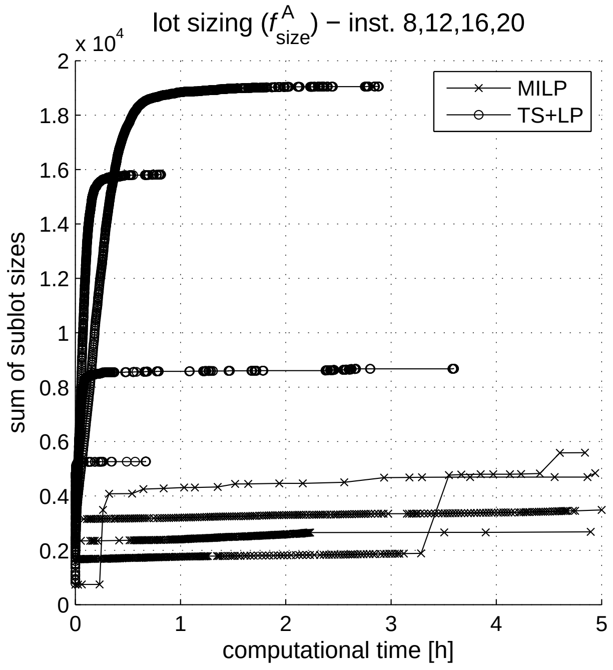
$$\Delta_6 = \frac{f^{\mathrm{B,GR(max)}}_{\mathrm{size}} - f^{\mathrm{B,GR(min)}}_{\mathrm{size}}}{f^{\mathrm{B,GR(max)}}_{\mathrm{size}}}, \quad \Delta^{\mathbf{B}}_{\mathbf{A}} = \frac{f^{\mathrm{B,MILP-MM}}_{\mathrm{size}} - f^{\mathrm{B,best(A)}}_{\mathrm{size}}}{f^{\mathrm{B,MILP-MM}}_{\mathrm{size}}}, \quad \rho = \frac{\mathrm{val}}{|\mathbf{O}|},$$

val, UB, gap – as mentioned for Table 4, ν – number of solutions generated per second,

$\theta_{\max}$ – percentage of solutions with the best objective value in the set of all solutions,



(a)



(b)

Figure 4: Objective function value as a function of time for selected instances, methods and criteria

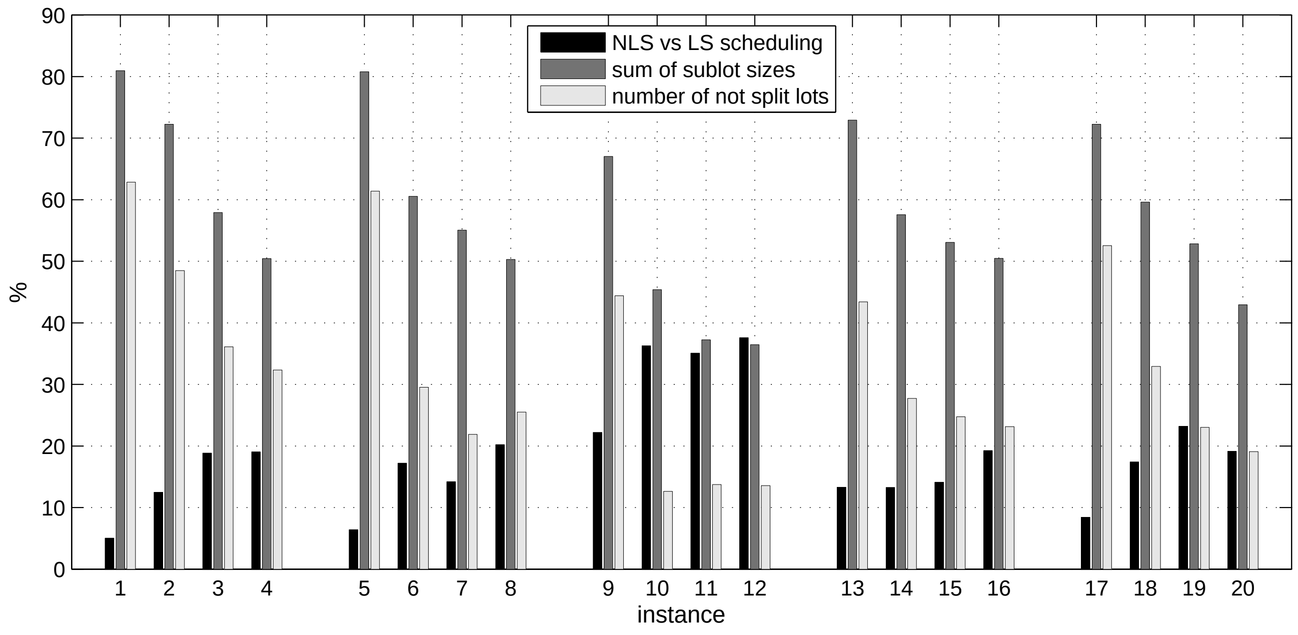


Figure 5: Comparison of the main parameters across the benchmark instances

- the best (maximum) value of the sum of the sublot sizes $f^{\mathrm{A}}_{\mathrm{size}}$ (Table 4) normalized with respect to the bounds given in Table 1, i.e., equal to $(f^{\mathrm{A}}_{\mathrm{size}} - \sum d_{k,i})/(\sum e_k - \sum d_{k,i})$, $k \in \{1, \ldots, n\}$, $i \in \{1, \ldots, o(k)\}$,
- the number of non-split lots relative to the number of operations, expressed by the parameter ρ of MILP-MM (Table 5).

One can expect that the difference in the makespan values between the NLS and LS solutions should grow with the number of sublots per operation, whereas the sum of the sublot sizes and the number of non-split lots should decrease at the same time. The bar plot confirms this assumption. The expected relationships are evident, although perturbed in some cases, because the benchmark instances have been generated randomly.

8 Summary

Several efficacious optimization approaches have been proposed for the considered two-stage lot streaming and lot sizing flexible job shop problem. The approaches relate to three objective functions and include implementations based on third-party proprietary solvers for the MILP and CP problem models, as well as alternative implementations, based mainly on a tabu search metaheuristic.

The use of a third-party solver supports an easy and rapid implementation, because only a declarative problem model is needed, with no explicit solution algorithm. Because of that, the approach is also very flexible, and details of the implementation can be quickly corrected and adjusted. Additionally, it is important from a research point of view that the MILP and CP approaches may provide a proof of optimality. Despite it is possible only for smaller problem instances, the solutions proven to be optimal are usable as references for testing alternative approaches.

On the other hand, the directly implemented metaheuristic algorithms have resulted in a better performance for the largest problem instances, in particular, the TS approach turned out to be better than CPO in the case of scheduling with lot streaming, as well as the TS+LP implementation of lot sizing has dominated the MILP-based approach. The possibility of a full control of the implementation details is another advance of these algorithms. For instance, the size of the neighborhood can be easily modified to tune the trade-off between the speed of the search in the solution space and the quality of the solutions.

The combination of the flexible job shop with lot sizing using variable sublots is the primary contribution of this work. The lot streaming is usually considered in combination with the flow shop in the related works. Moreover, even if the job shop is extended by lot streaming, the simultaneous sizing of sublots is very rarely analyzed. The sizing of variable sublots is especially seldom, even for the flow shop problem, because complex models are needed. In this work, a graph and MILP models of the considered sublot sizing problem have been introduced, together with a complete and verified implementation of optimization algorithms. The use of variable sublots is crucial in this work. First, this makes it possible to optimize the sublot sizes effectively, keeping the value of the objective function of the scheduling stage not worsened. Second, it is the base of the practical usefulness of the proposed solutions in the case of a manufacturing organization in which sublot sizes differ for individual processes.

The presented approaches can be easily adapted to a broader set of problems. The two stages of optimization are loose coupled, and the first stage (i.e., scheduling) can be formulated as a problem different from the flexible job shop with an objective function different from the makespan. The changes propagated to the second stage are slight in such a case, especially if a regular optimization criterion is used again. It is also worth to notice that some of the operations can be excluded from the procedure of lot sizing and can maintain a fixed sublot size if such a specific size is required in the manufacturing organization. Finally, specific conditions can be imposed by a simple modification of the domains of the decision variables which represent the lot sizes in the (MI)LP models, for instance, the sublot sizes can be multiplies of some minimum base size or may belong to a predefined set, representing the sizes of the containers available in a manufacturing system.

Acknowledgment

The research was supported by Short-Term Grant 2016/57214227 from Deutscher Akademischer Austauschdienst (DAAD). The authors would like to express their gratitude to Prof. Marian Wysocki (Department of Computer and Control Engineering, Rzeszow University of Technology, Poland) for his help in the formulation of the main assumptions of the work and many constructive comments.

References

[1] Alvarez-Valdez, R., Fuertes, A., Tamarit, J. M., Giménez, G., and Ramos, R., 2005. A heuristic to schedule flexible job-shop in a glass factory. *European Journal of Operational Research*, 165 (2), 525–534.

[2] Biskup D., and Feldmann M., 2006. Lot streaming with variable sublots: an integer programming formulation. *Journal of the Operational Research Society*, 57 (3), 296—303.

[3] Bożek A., and Wysocki M., 2015. Flexible job shop with continuous material flow. *International Journal of Production Research*, 53 (4), 1273–1290.

[4] Bożek A., and Wysocki M., 2016. Off-line and dynamic production scheduling – a comparative case study. *Management and Production Engineering Review*, 7 (1), 21–32.

[5] Calleja G., and Pastor R., 2014. A dispatching algorithm for flexible job-shop scheduling with transfer batches: an industrial application. *Production Planning & Control*, 25 (2), 93–109.

[6] Chan F. T. S., Wong T. C., and Chan P. L. Y., 2004. Equal size lot streaming to job-shop scheduling problem using genetic algorithms. *IEEE International Symposium on Intelligent Control*, Taipei, 2-4 September 2004, 472–476.

[7] Chan F. T. S., Wong T. C., and Chan L. Y., 2008. Lot streaming for product assembly in job shop environment. *Robotics and Computer-Integrated Manufacturing*, 24 (3), 321–331.

[8] Chen M., 2007. Lot Sizing at the Operational Planning and Shop Floor Scheduling Levels of the Decision Hierarchy of Various Production Systems. PhD dissertation. Virginia Polytechnic Institute and State University.

[9] Cheng M., Mukherjee N. J., and Sarin S. C., 2013. A review of lot streaming. *International Journal of Production Research*, 51 (23-24), 7023–7046,

[10] Defersha F. M., and Chen M., 2010. A hybrid genetic algorithm for flowshop lot streaming with setups and variable sublots. *International Journal of Production Research*, 48 (6), 1705–1726.

[11] Defersha F. M., and Chen M., 2012. Mathematical model and parallel genetic algorithm for hybrid flexible flowshop lot streaming problem. *International Journal of Advanced Manufacturing Technology*, 62 (1), 249–265.

[12] Defersha F. M., and Chen M., 2012. Jobshop lot streaming with routing flexibility, sequence-dependent setups, machine release dates and lag time. *International Journal of Advanced Manufacturing Technology*, 50 (8), 2331–2352.

[13] Demir Y., and İşleyen S. K., 2014. An effective genetic algorithm for flexible job-shop scheduling with overlapping in operations. *International Journal of Production Research*, 52 (13), 3905–3921.

[14] Edabi A., and Moslehi G., 2013. An optimal method for the preemptive job shop scheduling problem. *Computers & Operations Research*, 40 (5), 1314–1327.

[15] Edis R. S., and Ornek M. A., 2009. Simulation analysis of lot streaming in job shops with transportation queue disciplines. *Simulation Modelling Practice and Theory*, 17 (2), 442—453.

[16] Edis R. S., and Ornek M. A., 2009. A tabu search-based heuristic for single-product lot streaming problems in flow shops. *International Journal of Advanced Manufacturing Technology*, 43, 1202–1213.

[17] Fattahi, P., Jolai, F., and Arkat J., 2009. Flexible job shop scheduling with overlapping in operations. *Applied Mathematical Modelling*, 33 (7), 3076–3087.

[18] Gearhart J. L., Adair K. L., Detry R. J., Durfee J. D., Jones K. A., and Martin N., 2013. Comparison of open-source linear programming solvers. Sandia Report, Sandia National Laboratories, SAND2013-8847.

[19] Gröflin H., Pham D. N., and Bürgy R., 2011. The flexible blocking job shop with transfer and set-up times. *Journal of Combinatorial Optimization*, 22 (2), 121—144.

[20] Jungwattanakit J., Reodecha M., Chaovalitwongse P., and Werner F., 2008. Algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. *International Journal of Advanced Manufacturing Technology*, 37 (3), 354–370.

[21] Jungwattanakit J., Reodecha M., Chaovalitwongse P., and Werner F., 2009. A comparison of scheduling algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. *Computers & Operations Research*, 36 (2), 358–378.

[22] Jun-Jie B., Yi-Guang G., Ning-Sheng W., and Dun-Bing T., 2009. An Improved PSO Algorithm for Flexible Job Shop Scheduling with Lot-Splitting. *International Workshop on Intelligent Systems and Applications*, Wuhan, 23-24 May 2009, 1–5.

[23] Karimi B., Ghomi S. M. T. F., and Wilson J. M., 2003. The capacitated lot sizing problem: a review of models and algorithms. *Omega – The International Journal of Management Science*, 31 (5), 365–378.

[24] Lei D., and Guo X., 2013. Scheduling job shop with lot streaming and transportation through a modified artificial bee colony. *International Journal of Production Research*, 51 (16), 4930–4941.

[25] Liu S. C., 2003. A heuristic method for discrete lot streaming with variable sublots in a flow shop. *International Journal of Advanced Manufacturing Technology*, 22 (9), 662–668.

[26] Low C., Hsu C. M., and Huang K. I, 2004. Benefits of lot splitting in job-shop scheduling. *International Journal of Advanced Manufacturing Technology*, 24 (9), 773—780.

[27] Meindl B., and Templ M., 2012. Analysis of commercial and free and open source solvers for linear optimization problems. Research report. Department of Statistics and Probability Theory, Vienna University of Technology.

[28] Mortezaei N., and Zulkifli N., 2014. A study on integration of lot sizing and flow shop lot streaming problems. *Arabian Journal for Science and Engineering*, 39 (12), 9283–9300.

[29] Pan Q. K., and Ruiz R., 2012. An estimation of distribution algorithm for lot-streaming flow shop problems with setup times. *Omega*, 40 (2), 166-–180.

[30] Petrovic S., Fayad C., Petrovic D., Burke E., and Kendall G., 2008. Fuzzy job shop scheduling with lot-sizing. *Annals of Operations Research*, 159 (1), 275–292.

[31] Potts C. N., and Baker K. R., 1989. Flow shop scheduling with lot streaming. *Operations Research Letters*, 8 (6), 297–303.

[32] Rohaninejad M., Kheirkhah A., and Fattahi P., 2015. Simultaneous lot-sizing and scheduling in flexible job shop problems. *International Journal of Advanced Manufacturing Technology*, 78 (1), 1–18.

[33] Rossi A., 2014. Flexible job shop scheduling with sequence-dependent setup and transportation times by ant colony with reinforced pheromone relationships. *International Journal of Production Economics*, vol.153, 253–267.

[34] Sarin S. C., and Jaiprakash P., 2007. Flow Shop Lot Streaming, Springer US.

[35] Tseng C. T., and Liao C. J., 2008. A discrete particle swarm optimization for lot-streaming flowshop scheduling problem. *European Journal of Operational Research*, 191 (2), 360—373.

[36] Ventura J. A., and Yoon S.-H., 2013. A new genetic algorithm for lot-streaming flow shop scheduling with limited capacity buffers. *Journal of Intelligent Manufacturing*, 24 (6), 1185—1196.

[37] Werner F., 2013. A survey of genetic algorithms for shop scheduling problems. [in:] Siarry P. (ed.), *Heuristics: Theory and Applications*, Nova Science Publishers, 161–222.