The Multiple Batch Processing Machine Problem with Stage Specific Incompatible Job Families

Marc-Andrè Isenberg and Bernd Scholz-Reiter

Abstract This contribution introduces the problem of batching and scheduling n jobs in non-permutation flow shops with $m \geq 2$ batch processing machines in sequence and stage specific incompatible job families. Batches are build at each stage according to family-specific parameters. This creates a stage-interdependent batching and scheduling problem. We formulate the model by considering different options in terms of release dates, due dates, objective functions as well as setup times and define a corresponding integer linear program.

Introduction

In this contribution we define the multiple batch processing machine problem with stage specific incompatible job families ($MBPM_{SIF}$) and test its complexity factor. This problem consist of batching and scheduling multiple products in flow shop environments with multiple stages and intermediate storages in between. In general we understand batching as the task of selecting jobs and building batches out of jobs. Scheduling generates machine specific batch sequences by determining starting and ending times of batch productions. Each stage has one batch processing machine (BPM) with limited capacity. The product orders are represented by jobs during the production phase. All jobs have to be processed on all machines in the same technological order. However, the job sequence is allowed to change between the stages. This is called a *non-permutation schedule*. For changing the job sequence and for temporary storage we take intermediate storages with infinite capacities into our

M.-A. Isenberg (✉) · B. Scholz-Reiter
Planning and Control of Production Systems, University of Bremen,
Hochschulring 20 D 28359 Bremen, Germany
e-mail: ise@biba.uni-bremen.de

H.-J. Kreowski et al. (eds.), *Dynamics in Logistics*, Lecture Notes in Logistics,
DOI: 10.1007/978-3-642-35966-8_9,

model consideration. Additionally, we include transport times from machine to storage and vice versa. We assume the options of common or distinct due dates. Furthermore, we introduce stage specific release dates which are the maximum of the *availability* and *realisability* of jobs. A job is available in a stage if it has completed all previous stages. A job is realisable if all materials which are required for stage specific job processing are available. The realisability options contain the material availability at *time zero* as well as *randomly-distributed* which describes the usage of an arbitrary distribution. There is full knowledge about the production demand and the resulting jobs for a specified time period. The time period can be divided into time slots. The jobs belong to *stage specific job families* with family-specific processing times. They are incompatible to each other. Such incompatibility results from the production of multiple products and considers realistic production scenarios where products have several product variants. Since product variants have partially the same components or colours jobs for different products can build common batches within a certain stage. This causes a reduction of the variety of different products to a stage specific number of different job families. The feature combination of multiple BPM in sequence and stage specific job families lead to *inconsistent batches* thus the job constellation of batches only persist for single stages. This is contrary to other batching and scheduling problems where the batch is only build once at the first stage and then endures. Furthermore, the batch building of jobs follows the *batch availability model* with *anticipatory setups*, *batch sequence-dependent setup-times* and a *non-preemptive scheduling* procedure [compare to (Allahverdi 2008; Potts and Kovalyov 2000)]. Following the batch availability model the jobs have the same starting and end time and become available for subsequent stages at once. Opposing to this in the job availability model jobs of a batch are independent from each other in terms of their completion times. Anticipatory setups specify the knowledge about the next batch in queue for a machine to arrange the setup before its arrival. Batch sequence-dependent setup-times appear if the previous batch on a machine belongs to a different family than the subsequent one. Non-preemptive scheduling exclude the possibility of setting manufacturing priorities for individual jobs. Since the $MBPM_{SIF}$ induces stage-interdependencies in batching and scheduling, reasonable objective functions are the minimisation of total flow time, total completion time or makespan. Following the notations of Graham et al. (1979), Potts and Kovalyov (2000), we denote our model as $Fm/s_{ifg}, b_i, d_j/\sum f_j$ (Graham et al. 1979; Potts and Kovalyov 2000). For more detailed descriptions about notations and characteristics we recommend (Allahverdi 2008; Potts and Kovalyov 2000).

The research is inspired by a real word scenario. Automakers partially ships their export cars in the so-called Completely-Knocked-Down procedure. Completely-Knocked-Down describes the shipment of cars as assembly sets to save import taxes in the country of destination. Within the preparation phase of the shipment the car components are picked and packed into boxes. To gain economies of scale this process is done by building batches out of boxes with the same content. Following, the boxes come into an intermediate storage. Subsequent the boxes are stuffed into containers with different destinations. The transfer to the

above model works as follows: the *picking and packing process* is the first BPM, the *stuffing process* the second one. The creation of batches derives from clustering boxes with identical content and stuffing boxes with equal destinations into the same container. Due to this exigences both BPM have their own job families. Also they are limited in space and volume thus they have a maximum batch size.

The objective of this contribution is the formulation of the $MBPM_{SIF}$ achieving a comprehensive understanding of the problem and thus enabling the development of solution approaches. The article is structures as follows: section Related Work presents a literature review concerning related work. section Problem Statement illustrates the problem and formulates the mathematical model and integer linear program respectively. A summary of the contribution as well as an outlook are presented in section Summary and Outlook.

Related Work

There exist a number of reviews and surveys which precis the state of the art for research concerning batching and scheduling (Allahverdi 2008; Mathirajan and Sivakumar 2006; Potts and Kovalyov 2000; Webster and Baker 1995). We started our literature research on the basis of their categorizations and went over to more specified criteria of our problem such as *multiple batch processing machines in sequence* and *stage specific incompatible job families*. These characteristics are the most important ones since they cause discontinues processing and disjunctive sets of jobs for batch formation thus increasing the complexity of batching and scheduling. Furthermore, we only considered batch availability models and offline-algorithm. The focus on offline-algorithm derives from our model which underlies full knowledge of the production demand of a future period. That induces deterministic behaviour.

The combination of multiple BPM in sequence and stage specific job families automatically leads to the appearance of *inconsistent batches*. There is a very limited number of papers which explicitly consider this characteristic. Ng and Kovalyov (2007) study flow shops with several BPM in sequence and machine-dependent setup-times. They prove for several problems that a permutation schedule is the optimal one (Ng and Kovalyov 2007). However, their demonstration only includes compatible job families. The characteristic of stage specific incompatible job families would impede the feasibility of a permutation schedule. Buscher and Shen (2009) develop a mixed integer programming formulation implying inconsistent batches for a flow shop scheduling problem involving several BPM. Their program is only able to solve small instances and does not consider incompatible job families. Additionally, there are some more approaches on flow shops with multiple BPM in sequence composing consistent batches. For instance Kumar Manjeshwar et al. (2009) as well as Lei and Guo (2011) apply meta-heuristics such as simulated annealing and variable neighbourhood search (Kumar Manjeshwar et al. 2009; Lei and Guo 2011).

Multiple stages and incompatible job families are considered by Fu et al. (2011), Kim et al. (2001), Tajan et al. (2006). Fu et al. (2011) analyse the problem of a two-stage flow shop containing a single BPM following by a discrete processor and a limited buffer in between Fu et al. (2011). Kim et al. (2001) as well as Tajan et al. (2006) study the reverse order which consists of a serial processor feeding a BPM (Yd et al. 2001; Tajan et al. 2006). In all models the incompatible job families are only valid for the batch processing stage. Besides the flow shop focus there are several papers which concentrate on batching and scheduling of single batch processing machines regarding incompatible job families ($SBPM_{IF}$). Yao et al. (2012) consider dynamically arriving jobs and analyse different dominance properties. They deduce several lower bounds and propose a basic branch-and-bound algorithm, which they improve to a decomposed branch-and-bound algorithm regarding the dynamic job arrival. The objective is the minimisation of the total completion time (Yao et al. 2012). Koh et al. (2005) study the $SBPM_{IF}$ in a multi-layer ceramic capacitor production line and propose a number of heuristics as well as hybrid genetic algorithm to solve problems with an extensive number of jobs in reasonable computation time (Koh et al. 2005). A potential transfer of the $SBPM_{IF}$ approaches to our model is in adapting them onto each stage of our model. However, the stages would be planned separately and without any consideration of interdependencies out of the stage specific scheduling.

Additionally, many researchers analyse the batching and scheduling problems of single stages including identical or non-identical parallel batch processing machines with incompatible job families ($PBPM_{IF}$).Their research is mostly motivated by semiconductor wafer fabrication. Jula and Leachman (2010) propose exact and heuristic algorithms for the $PBPM_{IF}$. Their model also includes feeding and removing apparatus (Jula and Leachman 2010). Chiang and Cheng (2010) develop a memetic algorithm for minimizing total weighted tardiness in $PBPM_{IF}$ regarding dynamic job arrival and incompatible job families. Their algorithm plans the batch formation and batch sequence simultaneously (Chiang and Cheng 2010). Since the papers of this category mainly focus on the usage of common resources and do not regard scheduling dependencies across stages we do not consider more approaches and refer to Mathirajan and Sivakumar (2006).

Summarising the results of our literature research we notice that no analysed model or approach combines the characteristics of *multiple batch processing machines in sequence* and *stage specific incompatible job families*. However, the illustrated research presents related and interesting work and deals with certain elements of our problem. Hence, they may support the following problem formulation.

Problem Statement

In general, batching and scheduling problems consist of two fundamental decision for each BPM. The first one concerns the batch formation, the second one contains the batch sequencing. Regarding our model we need to be aware that both

problems are related to each other due to the characteristics such as incompatible job families and BPM capacities. Naturally the batch formation precedes. However, it influences the completion time of a batch by arranging its batch size and thus determining the availability of the containing jobs for subsequent BPM. Since the following BPM has another job families than its predecessor the batch cannot automatically proceed as a whole, it could contain incompatible families for the BPM. This effects a necessary re-formation of the batches between the stages. Smaller batch sizes could reduce this problem, but they are causing more setup-times by an increased changing rate of batch families and decrease the efficiency of the BPM. For permuting the job sequence and composing new batch formations there are storages with unlimited capacities located between the BPM. This ensures that BPM never block. Due to the option of distinct due dates and the re-formation of batches we notice that jobs of the same batch may vary in their stage specific release dates and deadlines. In addition, the planning problem could be more complex by introducing production time slots. They have limited time capacities and batches are mostly not allowed to split across them. They need to be completed within one time slot. However, the option of different time slots are not considered in the following notation and problem formulation.

Mathematical Problem Formulation

Following, we present the notation and underlying assumptions of the integer linear program of the $MBPM_{SIF}$. We use the following notation:

Subscripts

$k = 1,\ldots,m$	Machine index
$j, i, c = 1,\ldots,n$	Job index
$s = 1,\ldots,x$	Storage index
$a_k = 1,\ldots,g_k$	Family index on machine k

Parameters

$P_{k,j}$	Processing time of job j on machine k
Z_j	Due date of job j
$u_{k,j}$	Setup time for job j on machine k (equal for all family members)
$SMIN_k$, $SMAX_k$	Minimum and maximum batch size at machine k
$T_{k,s}$, $T'_{s,k}$	Transport time for batches from k to s and vice versa
$EoS_{k,j}$	Economies of scale for job j if its batch has more than one job
$bigM$	Big integer number

Binary variables

$\delta_{k,j,i}$	1 if job j is scheduled before job i on machine k
$\theta_{k,j,i}$	1 if job j is direct predecessor of job i on machine k
$\gamma_{k,j,i}$	1 if job j and i belongs to the same family on machine k
$\pi_{k,j}$	1 if job j has a directly preceding job on machine k which belongs to the same family
$B_{k,j,i}$	1 if job j and job i belongs to the same batch on machine k
$\phi_{k,j,i,c}$	1 if job c is scheduled between j and i on machine k

Integer variables

C_j	Completion time of job j
f_j	Flow time of job j
C_{max}	Makespan
$r_{k,j}$, $r'_{k,j}$	Start and completion time of job j on machine k
$v_{k,j}$	Shortfall below $SMIN_k$ of the batch containing job j
$D_{k,j}$	Deadline of job j on machine k
$A_{k,j}$	Realisability date of job j on machine k
$R_{k,j}$	Release date of job j on machine k

There are several assumptions. Besides the intermediate storages additional storages are located before the first and after the last batch processor. Hence, the flow shop has $x = m + 1$ storages. All jobs have to pass all storages. Transport times are from machines to storages and vice versa and count for the whole batch. The production planning is far ahead of the end of the production period thus we assume an infinite time horizon. The following integer linear program focuses mainly on the minimisation of the total flow time. Thereby it uses batch sequence-dependent setups. Both can be easily substituted by other objectives and setup approaches.

Problem Formulation

Objective function

$$min\ v_{k,j} + \begin{cases} \sum_{j=1}^{n} f_j & \text{if objective is total flow time} \\ \sum_{j=1}^{n} C_j & \text{if objective is total completion time} \\ C_{max} & \text{if objective is makespan} \end{cases} \tag{1}$$

Environmental constraints and definition

$$f \geq \sum_{k=1}^{m} \sum_{j=1}^{n} v_{k,j} + \sum_{j=1}^{n} (Z_j - r_{1,j}) \tag{2}$$

$$r'_{k,j} \geq r_{k,j} + (1 - \pi_{k,j}) \cdot u_{k,j} + P_{k,j} \cdot \sum_{i=1}^{n} B_{k,j,i} - EoS_{k,j} \cdot (-1 + \sum_{i=1}^{n} B_{k,j,i}) \quad \forall k, j \quad (3)$$

$$\sum_{j=1}^{n} \theta_{k,j,i} \cdot \gamma'_{k,j,i} - \pi_{k,i} \cdot bigM \leq 0 \quad \forall k, i \quad (4)$$

$$\sum_{j=1}^{n} \theta_{k,j,i} \cdot \gamma'_{k,j,i} + (1 - \pi_{k,i}) \cdot bigM \geq 1 \quad \forall i \quad (5)$$

$$r_{k,j} \geq r'_{k-1,j} + T_{k-1,s} + T'_{s,k} \quad \forall k > 1, j, s \quad (6)$$

$$r'_{m,j} + T_{m,x} \leq Z_j \quad \forall j \quad (7)$$

Machine constraints

$$\sum_{j=1}^{n} B_{k,j,i} \leq SMAX_k \quad \forall k, i \quad (8)$$

$$v_{k,j} + \sum_{j=1}^{n} B_{k,j,.i} \geq SMIN_k \quad \forall k, i, v_{k,j} \leq 1 \quad (9)$$

$$r_{k,j} \geq r'_{k,i} - bigM \cdot (\delta_{k,j,i} + B_{k,j,i}) \quad \forall k, j, i, j \neq 1 \quad (10)$$

$$r_{k,i} \geq r_{k,i} - bigM \cdot (1 - \delta_{k,j,i} + B_{k,j,i}) \quad \forall k, j, i, j \neq 1 \quad (11)$$

$$\delta_{k,j,i} + \delta_{k,i,j} + B_{k,j,i} \leq 1 \quad \forall k, j, i \quad (12)$$

$$\delta_{k,j,i} + \delta_{k,j,c} + \delta_{k,c,i} - \phi_{k,j,i,c} \cdot bigM \leq 2 \quad \forall k, j, i, c \quad (13)$$

$$\delta_{k,j,i} + \delta_{k,j,c} + \delta_{k,c,i} + (1 - \phi_{k,j,i,c}) \cdot bigM \geq 3 \quad \forall k, j, i, c \quad (14)$$

$$\sum_{c=1}^{n} \phi_{k,j,i,c} - (1 - \delta_{k,j,i} + \theta_{k,j,i}) \cdot bigM \geq 1 \quad \forall k, j, i, j \neq i \quad (15)$$

$$\sum_{c=1}^{n} \phi_{k,j,i,c} - (2 - \delta_{k,j,i} - \theta_{k,j,i}) \cdot bigM \leq 0 \quad \forall k, j, i, j \neq i \quad (16)$$

$$\delta_{k,j,i} \geq \theta_{k,j,i} \quad \forall k, j, i \quad (17)$$

Batch constraints

$$r_{k,j} \geq r_{k,i} + 1 - (1 - \delta_{k,j,i}) - bigM \quad \forall k, j, i \quad (18)$$

$$r_{k,j} \geq r_{k,j} - \delta_{k,i,j} \cdot bigM \quad \forall k, j, i \quad (19)$$

$$B_{k,j,i} \leq \gamma'_{k,j,i} \quad \forall k,j,i \tag{20}$$

$$B_{k,j,j} \geq 1 \quad \forall k,j \tag{21}$$

$$\delta_{k,j,i} + \delta_{k,i,j} + B_{k,j,i} \cdot bigM \geq 1 \quad \forall k,j,i \tag{22}$$

The objective function (1) includes the options of minimising the total flow time, the total completion time or the makespan. Additionally, there are also the conceivable options of minimising the mean flow time $n^{-1}\sum f_j$ or mean completion time $n^{-1}\sum C_j$. Since they are equal to total flow time and total completion time we do not formulate them explicitly. The choice of the objective criterion depends on the motivation for analysing the MBPM_{SIF}. While the usage of the total flow time criterion allows a more intensive consideration of storages, the total completion time criterion focuses on the BPM. As the Eq. (2) describes, the above formulation focuses on the total flow time. However, the exclusively pursuance of this objectives would partially in contrast to the fundamental idea of batch processing which is driven by economic efficiency. Supposing the abandonment or insignificance of setup-times decision-makers would prefer batches of size one since they do not cause waiting times in storages until other jobs of the same job family are available. Hence, we introduce a weak constraint (9) for the minimal batch size of a BPM. This weak constraint is regarded in the objective function by the term of $v_{k,j}$ which defines the shortfall of the minimal batch sizes. It is restricted to allow a shortfall of at most 1. In contrary to Eq. (9) constraint (8) refers to the maximum number of jobs within a batch. Due to the physical bounded space it is constructed as a hard restriction which needs to be regarded strictly. The batch sequence-dependent setup events are determined by the constraints (4) and (5). These constraints work as an *either-or-option*. They analyse if one of the directly preceding jobs of j belongs to the same family as j does. The term (3) describes the completion time of a job depending on its production start on the machine, a potentially setup as well as the duration of the corresponding batch. The batch duration derives from the sum of the jobs processing times on machine k as well as on the economies of scale. This effect allegorises the savings of time which occur by the aggregation of material provisions, identical working procedures, learning effects etc. We advert that this formulation of setups only specifies *when* a setup precedes a batch, it does not quantify their duration. These could be constant for all families, family-dependent, machine-dependent or both. Constraint (6) defines the technological order of machines. All jobs needs to pass all machines and all storages. Equation (7) ensures that all jobs are in the last storage until their due dates. The constraints (10) and (11) identify the order of jobs and avoid overlapping of batches. They are also formulated as a *either-or-option*. In the case that job j is scheduled at any time before job i $\delta_{k,j,i}$ is *1*. If the jobs build a common batch or if j is scheduled after i $\delta_{k,j,i}$ is 0. Equation (12) avoids bidirectional hierarchies and ensures that two jobs either have one hierarchy proportion or that they constitute a common batch. The complementary constraint to this is (22). For the case that there is no hierarchy proportion it reasons that the jobs build a

common batch. The constraints (13 and 14) determine whether a job c is between job j and job i. This serves as a basis for the Eqs. (15 and 16). They count the number of jobs between job j and job i. If the sum of the intermediate jobs is 0 job j is a direct predecessor of i. In this case the sequence variable $\theta_{k,j,i}$ is set to 1. Constraint (17) defines that the sequence variable $\theta_{k,j,i}$ only can be set to 1 if the order variable $\delta_{k,j,i}$ also is 1. That means that job j can only be a predecessor of job i if they have an equal hierarchical order. The Eqs. (18 and 19) defines the same starting time for jobs of the same batch. The constraint (20) makes sure that jobs only can constitute a batch if they belong to the same job family on machine k. Equation (21) completes the matrix of $B_{k,j,i}$ and ensures that the belongings of two jobs to the same batch are bidirectional.

Analysing the problem size and complexity of the integer linear program we applied the pre-solver of GUROBI (Bixby et al. 2012) The scenario consist of $m = 2$ BPM, random job family processing times $P_{k,j}$ between 20 and 60 min, *EoS* is 15 % of the job′s process duration, the due dates Z_j out of an interval of 100 min, family-specific setup times $u_{k,j}$ of 50 % of a full sized batch of the specific family and transport times $T_{k,s}$ as well as $T'_{s,k}$ of 5 min. The other parameters $SMIN_k$, $SMAX_k$, g_k as well as the numbers of integer variables, binary variables and nonzeros after pre-solving are described for exemplary cases in Table 1.

Extending the above described model we further present the mathematical formulations of stage specific release dates and deadlines. They are necessary for the development of heuristically solution approaches solving the $MBPM_{SIF}$.

$$R_{k,j} = \begin{cases} \max\left\{A_{k,j}, r'_{k-1,j} + T_{k-1,s} + T'_{s,k}\right\} & \text{if } k > 1 \\ A_{j,k} & \text{else} \end{cases} \quad \forall k, s = k \tag{23}$$

$$D_{k,j} = \begin{cases} r_{k+1,j} + T'_{s,k+1} - T_{s,k} & \text{if } k < m \\ Z_j - T_{k,s} & \text{else} \end{cases} \quad \forall k, s = k \tag{24}$$

According to Eq. (23) the release date of job j on machine $k + 1$ results out of the maximum of its realisability date and its availability date. The realisability date corresponds to the provision of required material in the specific stage. The job availability date results from the completion of its batch on the previous machine $k - 1$ and the transport times for the batch from machine $k - 1$ to storage $s = k$ to

Table 1 Exemplary problem sizes

n	g_1	g_2	$SMIN_1$	$SMAX_1$	$SMIN_2$	$SMAX_2$	Integers	Binaries	Non-zeros
10	3	4	3	4	2	3	1,765	1,721	15,924
15	3	4	3	4	2	3	6,348	6,273	61,761
20	5	4	3	4	3	5	15,166	15,046	147,992
30	5	4	3	4	3	5	52,089	51,909	513,344
50	9	6	3	4	3	5	25,3847	243,547	2,427,108
60	9	6	3	4	3	5	42,3063	422,703	4,223,142

machine k. If $k = 1$ the availability of the job only depends on its realisability. The deadline $D_{j,k}$ of a job j for machine k results by calculating the reverse order in (24). If $k = m$ the job′s deadline derives from its due date instead of its starting time on the following machine. This calculations are equal if the $MBPM_{SIF}$ contains common or distinct due dates or if the realisability dates are chosen as time zero or distributed. If the focus of analysing the $MBPM_{SIF}$ excludes storages the transport times can be set to zero.

Problem Complexity

Since we allow a high variety of combinations in terms of due dates, deadlines, objective function etc. we only address exemplary cases for providing different perspectives on the complexity of $MBPM_{SIF}$. First we want to consider the case that we have different time slots with equal time capacity. The batches have non-identical durations. If we pursuit the minimisation of makespan C_{max} it is advantageous to realise a maximum usage of the time slots, i. e. no idle times of machines. This problem layout is equal to the one dimensional bin packing problem where we have bins of limited capacity and boxes of non-identical sizes. The bin packing problem is known as NP-hard (Garey et al. 1976). However, this reduction only regards the scheduling problem and does not work if we have only one time period with infinite time capacity. For this case Glass et al. (2001) proof for a flow shop of $m = 2$, $g_k = 1$ for all k, inconsistent batches and the objective of minimising the makespan that it is NP-hard in the strong sense. But they do not explicitly regard incompatible families. This is done by Webster and Baker (1995) for $SBPM_{IF}$ and different objectives. According to Uzsoy (1994) and for the objective of minimising total flow time they show that the problem is NP-hard (Webster and Baker 1995). The case of distinct release dates is considered by (Yuan et al. 2006). They proof that the $SBPM_{IF}$ is strongly NP-hard even for two distinct release dates and also if the processing times of the jobs are equal and the job families have identical setup-times. The $MBPM_{SIF}$ has distinct release dates in each stage of the flow shop. Garey and Johnson (1990) proved that finding the minimum mean flow time schedule in a flowshop of $m \geq 2$ machines is NP-complete (Garey et al. 1976). If we take the job constellation for batches as a given, the challenge of $MBPM_{SIF}$ only consist of finding the best schedule. Due to the presented results we assume that the $MBPM_{SIF}$ is also NP-complete. This degree of complexity is not effected by regarding or omitting distinct due dates or varying realisability dates. Different release dates in a stage results from the feature combination of multiple stages and stage specific incompatible job families. Therefore, all presented variants of the $MBPM_{SIF}$ are NP-hard.

Summary and Outlook

This contribution introduces and formulates the $MBPM_{SIF}$ as a logical extension of $SBPM_{IF}$ and $PBPM_{IF}$. The mathematical formulation describes a general model which allows flexibility in terms of the objective function, release dates, due dates, setup-times and storage consideration. Using a solver for model consolidation we illustrate the problem complexity for small scale test cases. According to the results of other researches we classify the $MBPM_{SIF}$ as NP-hard.

Further research should be done to find feasible solution approaches for the $MBPM_{SIF}$. Due to the hardness of the problem exact algorithms or MILP solver only can solve small instances. They could contribute to the understanding of the problem by determining the significance of single elements for the problem complexity. However, feasible solutions for more extensive instances only can be provided by heuristics. The development of heuristic needs to be made for specified characteristics of the $MBPM_{SIF}$ since this ensures a higher quality of the computed solution. Further one, such a solution could be serve as a basis method for improvement by developing and applying meta-heuristics [compare to Kumar Manjeshwar et al. (2009); Lei and Guo (2011)]. The $MBPM_{SIF}$ does not need restricted to the described options. Many more modifications and extensions are thinkable as long they do not change the characteristics of multiple stages and stage specific incompatible job families.

References

Allahverdi A (2008) A survey of scheduling problems with setup times or costs. Eur J Oper Res 187:985–1032

Bixby R, Gu Z, Rothberg E (2012) Gurobi. URL http://www.gurobi.com. Accessed 12/03/16

Buscher U, Shen L (2009) MIP-based approaches for solving scheduling problems with batch processing machines. Oper Res 132–139

Chiang T, Cheng H (2010) A memetic algorithm for minimizing total weighted tardiness on parallel batch machines with incompatible job families and dynamic job arrival. Comput Oper Res 37(12):2257–2269

Fu Q, Sivakumar A, Li K (2011) Optimisation of flow-shop scheduling with batch processor and limited buffer. Int J Prod Res p Forthcoming article the multiple batch processing machine problem 11

Garey MR, Johnson DS (1990) Computers and intractability; a guide to the theory of NP-completeness. W. H. Freeman & Co., New York

Garey MR, Johnson DS, Sethi R (1976) The complexity of flowshop and jobshop scheduling. Math Oper Res 1(2):117–129

Glass CA, Potts CN, Strusevich VA (2001) Scheduling batches with sequential job processing for two-machine flow and open shops. INFORMS J Comput 13(2):120–137

Graham RL, Lawler EL, Lenstra JK, Kan Rinnooy A (1979) Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann Discrete Math 5:287–326

Jula P, Leachman R (2010) Coordinated multistage scheduling of parallel batch-processing machines under multiresource constraints. Oper Res 58(4-Part-1):933–947

Kim Yd, Kim Jg, Choi B, Kim Hu (2001) Production scheduling in a semiconductor wafer fabrication facility producing multiple product types with distinct due dates. IEEE Trans Robot 17(5):589–598

Koh S, Koo P, Kim D, Hur W (2005) Scheduling a single batch processing machine with arbitrary job sizes and incompatible job families. Int J Prod Econ 98(1):81–96

Kumar Manjeshwar P, Damodaran P, Srihari K (2009) Minimizing makespan in a flow shop with two batch-processing machines using simulated annealing. Robot Comput Integrated Manuf 25(3):667–679

Lei D, Guo XP (2011) Variable neighbourhood search for minimising tardiness objectives on flow shop with batch processing machines. Int J Prod Res 49(2):519–529

Mathirajan M, Sivakumar A (2006) A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor. Int J Adv Manuf Tech 29(9–10):990–1001

Ng CT, Kovalyov MY (2007) Batching and scheduling in a multi-machine flow shop. J Sched 10(6):353–364

Potts C, Kovalyov M (2000) Scheduling with batching: a review. Eur J Oper Res 120(2):228–249

Tajan JBC, Sivakumar AI, Gershwin SB (2006) Performance of a serial-batch processor system with incompatible job families under simple control policies. Mechanical Engineering (65), URL http://hdl.handle.net/1721.1/29835

Uzsoy R (1994) Scheduling a single batch processing machine with nonidentical job sizes. Int J Prod Res 32(7):1615–1635

Webster S, Baker KR (1995) Scheduling groups of jobs on a single machine. Oper Res 43(4):692–703

Yao S, Jiang Z, Li N (2012) A branch and bound algorithm for minimizing total completion time on a single batch machine with incompatible job families and dynamic arrivals. Comput Oper Res 39(5):939–951

Yuan JJ, Liu ZH, Ng CT, Cheng TCE (2006) Single machine batch scheduling problem with family setup times and release dates to minimize makespan. J Sched 9(6):499–551