A heuristic for variable re-entrant scheduling problems

Citation for published version (APA):

Document status and date:
Published: 01/06/2018

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 29. Jul. 2019
A Heuristic for Variable Re-Entrant Scheduling Problems

Roel van der Tempel, Joost van Pinxten, Marc Geilen, and Umar Waqas
A Heuristic for Variable Re-Entrant Scheduling Problems

Roel van der Tempel*, Joost van Pinxten†, Marc Geilen† and Umar Waqas†
*Research and Development, Océ Technologies, Venlo, The Netherlands
†Department of Electrical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands

Abstract—Flexible Manufacturing Systems (FMSs) need a scheduler to provide timing instructions for the operations of different products. Previous work has presented heuristics for fixed-order 2-re-entrant scheduling problems [1], [2]; where products visit a re-entrant machine exactly two times for production. We propose an extension to this scheduling model, and an extension to the scheduling heuristic, that allows jobs to move along different flows on re-entrant machines; i.e. jobs can (re)visit the re-entrant machine once or twice. An FMS that requires such variable re-entrance with fixed-order output is a Large Scale Printer (LSP). The scheduling problem in an LSP is modeled as a variable re-entrance flowshop with relative due dates and sequence-dependent setup times. We identify the impact of the fixed order requirement on the variable re-entrance scheduling problem. We show that out-of-order input of products can be beneficial to the scheduling quality in variable re-entrance scheduling. The fixed re-entrant heuristic of [1] is extended with a method to order operations on the re-entrant machine to minimize the completion time of variable re-entrant job sets. The resulting heuristic produces good quality schedules for variable re-entrant job sets without losing schedule quality for fixed re-entrant job sets.

I. INTRODUCTION

In increasing market globalization, customers demand production systems with high flexibility and reconfigurability [4]. In these production systems, it is essential that the system can rapidly adapt and be configured to the customer needs. The printing market for example is a global market with high productivity demands. Production requests arrive only seconds before the start of the production process. Many modern day production printers are Flexible Manufacturing Systems (FMSs) that rapidly adapts to product requirements [5], [6].

A Large Scale Printer (LSP) is an FMS capable of printing hundreds of thousands of sheets per day in high quality. It can print various types of media on a single machine, where each individual sheet has customizable graphics. The machines in a printer need time to reconfigure, depending on the consecutive media types, i.e. they have machine flexibility [5], [6]. Some media are not allowed to re-enter a machine: tab sheets have flaps that would get damaged during transport in the return loop. An LSP must schedule the required sequence of operations to print sheets of a print job on-line.

In this paper we focus on the problem of scheduling operations of sheets inside an LSP. We provide a productive scheduling mechanism for print jobs with different media types and re-entrance requirements, i.e. different range flexibility and production flexibility [5], [6].

The LSP consists of a set of machines that each have a specific task in the printing process. The paths that sheets can follow through the different machines inside the LSP is modeled in Fig. 1. The arrows show the flow of sheets through the LSP. Sheets enter the printer at the Paper Input Machine (PIM), are printed at the Image Transfer Machine (ITM) and are unloaded at the Paper Output Machine (POM). The ITM takes a major share of the production costs. Duplication of the ITM is therefore prohibitively expensive. To print sheets either on one side or on both sides, sheets can follow two routes. A route consists of segments between machines where sheets travel from one machine to the next. In the simplex route sheets are loaded by the PIM, printed on their first side by the ITM, and directly sent to the POM to be unloaded. In the duplex route sheets are loaded by the PIM and printed on their first side by the ITM, referred to as a first pass. After the first pass, the re-entrant path is taken, a track on which sheets are turned upside down and re-enter the printer at the Merge Point. In this way the sheet is printed on its second side by the ITM, referred to as a second pass. Finally, the duplex sheet is unloaded by the POM. To summarize, a simplex job is printed on one side and a duplex sheet is printed on both sides. Combining sequences of simplex and duplex sheets in one print job is referred to as mixed-plexity.

An on-line scheduler inside an LSP must decide on the processing order of sheets that are processed on the same
machine. This is limited by several physical properties.

- The machines can process no more than a single sheet at a time. The duration that machines take to process a sheet is called the processing time.
- A machine may require additional steps before the next sheet is processed, referred to as setup time, for instance when consecutive sheets have different media types. The setup time can be derived from physical parameters such as sheet length and thickness. This makes the setup time between sheets sequence-dependent.
- The travel time between machines is determined by physical constraints on heating, cooling, sheet length, track acceleration and velocity. The sheets must continue to move on the track, and cannot be indefinitely buffered: i.e., the travel time of sheets between machines is not indefinite. This imposes due dates that indicate when the next machine in the flow must be ready to start processing a sheet. The track in the re-entrant path can be partially accelerated/decelerated, so that the travel time can be varied within bounds, creating a virtual buffer.
- Sheets cannot overtake other sheets while traveling on a segment from one machine to the next machine. Thus the order of sheets at the start of a segment is the same as the order of sheets at the end of the segment.
- Finally, sheets must be unloaded in a fixed order, as defined by the print job.

The scheduling freedom is therefore deciding the order and timing of sheets coming from the PIM and from the re-entrant path. The LSP scheduler must provide a feasible schedule such that the above mentioned properties hold.

In this paper we focus on the scheduling problem where sheets can take the duplex and simplex route. Schedulers have been developed for the LSP [1], [2], but these approaches support only print jobs that are fully simplex or fully duplex. They cannot schedule mixed-plexity print jobs in an LSP, i.e. variable re-entrancy job sets.

We identify changes to the scheduling model for jobs with variable re-entrancy and discuss the new properties that arise. The scheduling mechanism of the Heuristic Constrained-based Scheduler (HCS) is extended for job sets that have variable re-entrancy. This version is called the Mixed-Plexity HCS (MPHCS). It introduces a method for variable re-entrancy scheduling and provides high quality schedules by allowing products to enter in a different order than the order in which they must leave the LSP.

The next section positions the mixed-plexity scheduling problem with scheduling problems from literature. Section III formally defines the scheduling problem as a flowshop variant. Section IV discusses the used method to model the scheduling problem. The extension to the scheduling mechanism of [1] is presented in section V. Section VI shows the experimental evaluation. In section VII the paper concludes.

II. RELATED WORK

Machines or factories that can produce (semi-)custom products are typically modelled by some variant of a jobshop or flowshop and scheduled by its associated algorithms. Many variants and classifications of jobshops exist. In this work, we are particularly interested in jobshop classes that have precedence constraints between jobs and operations of jobs. Jobshops with precedence constraints can take many forms [7], yet few deal with fixed job completion order [8]. Even fewer describe the effect of allowing or disallowing overtaking of other processed jobs [9], [10].

Flowshops define restrictions on the operation-order within a job, but do not typically restrict the order between operations of different jobs. In re-entrant flowshop scheduling, the flow, i.e. the order of operations in a job, has been fixed, but the scheduler is allowed to change the order of the jobs, as long as the given precedence constraints are not violated. For practical reasons, the resulting schedules are typically limited to permutation schedules, where the jobs are given an order and the operations all follow the same order. If the job order is fixed, then the scheduling freedom consists of interleaving different operations of different jobs [8]. The approach in [8] also allows variation in job types. After jobs have been processed by a shared common machine, each job type flows to a dedicated machine. Our work gives a method to schedule re-entrant flowshops with limited variability on the flows, and all job types are processed by a shared machine.

A re-entrant flowshop formulation with varying flows has been proposed in [11], [12]. Each job can have a different flow associated with it, possibly re-entering the same machine several times. However, this model allows arbitrary overtaking of jobs, and it does not impose any restrictions on the order of completion of the last operation of a job, nor are sequence-dependent setup times and relative due dates modelled. Due to the lack of these model elements, the solution proposed in [11] is not applicable to our use case.

Most solutions for (re-entrant) flowshops do not consider relative due dates or fixed job completion order and therefore do not take feasibility into account. Less research is done on jobshop and flowshop problems that involve relative due dates and sequence-dependent setup times. The work of [7] solves a jobshop scheduling problem with setup times, deadlines and precedence constraints by transforming into a Traveling Salesman problem with Time Windows (TSPTW). The approach cannot be trivially adjusted to relative due dates as the resulting TSPTW cannot handle relative time windows.

Schedulers for job sets with fixed flows (i.e. simplex, or duplex, but not mixed), have been proposed for the LSP in [1], [2]. The MFL and MNEH heuristics [13] are also able to schedule 2-re-entrant flowshops with sequence dependent setup times and relative due dates. They are however not fit for online scheduling of re-entrant flowshops [1] in terms of runtime and would require extensions to allow variable re-entrancy without allowing overtaking.

An approach using priced timed automata to model and schedule variable recipes (i.e. flows) on multiple resources for lacquer production is described in [14]. One of their heuristics describes that restricting to non-overtaking schedules is probably more productive when processing and setup times are very
similar, but the problem definition still allows overtaking. A straightforward transcription of the re-entrant flowshop model of [1] (which is the basis for our extension) into priced timed automata similar to [14] revealed significant scalability issues when relative due dates and setup times need to be taken into account, which makes it unfit for on-line scheduling.

We did not examine the variable flow re-entrant flowshop without overtaking any further, as the intended scheduling algorithm should be able to provide solutions on-line for fixed-flow and variable-flow product sequences with sequence-dependent setup times and relative due dates.

III. PROBLEM DEFINITION

We model the LSP using the following definitions.

Definition 1. A variable re-entrant flowshop with sequence-dependent setup times and relative due dates is a tuple \((M, J, \theta, r, O, P, S, D)\), according to the following definitions.

Definition 1.1. (Flowshop model) \(M = \{\mu_1, \ldots, \mu_m\}\) is the set of machines. The LSP has \(m = 3\) machines. \(J = \{j_1, \ldots, j_k\}\) denotes the sequence of jobs in completion order. In the LSP, the second machine \(\mu_2\) is a re-entrant machine. The function \(\theta : J \rightarrow M^*\) specifies the sequence of machines that are visited by a job to execute the operations. The sequence of a simplex job \(s\) is \(\theta(s) = \mu_1\mu_2\mu_3\). A duplex job has the sequence \(\theta(d) = \mu_1\mu_2\mu_1\mu_2\mu_3\). \(\theta(j)(i)\) shows on which machine the \(i\)-th operation is mapped in the sequence of a job \(j\). For every job \(j \in J\) a number of \(r(j) = |\theta(j)|\) operations are executed. I.e. for a duplex job \(r(d) = 4\) and for a simplex job \(r(s) = 3\). \(O = \{o_{a,v} | u \in J, 1 \leq v \leq r(u)\}\) is the set of operations that are executed for the jobs in \(J\).

Definition 1.2. (Timing requirements) The processing times to execute an operation by a machine are defined as \(P : O \rightarrow \mathbb{R}_{>0}\). The setup times contain two types of time delays. The first delay type is the minimum travel time between two operations of the same job \(j \in J\), which imposes a time delay between the finish of \(o_{j,i}\) and start of \(o_{j,i+1}\). The second delay type is the sequence-dependent time between two consecutively scheduled operations of jobs \(j_1, j_2 \in J\) on the same machine \(\mu \in M\). This delay is denoted by \(S : O \times O \rightarrow \mathbb{R}_{\geq 0}\), and \(S(o_{a,u}, o_{c,v})\) imposes a minimal delay between the finish of \(o_{a,u}\) and start of \(o_{c,v}\). Relative due dates are maximum delays between the starting times of two operations, denoted by \(D : O \times O \rightarrow \mathbb{R}_{\geq 0}\). In this work relative due dates occur only within a job. \(D(o_{a,j}, o_{a,j+1})\) is the maximum delay between the finish of \(o_{a,j}\) and start of \(o_{a,j+1}\).

Definition 2. (Constraints) A schedule \(\Omega\) for the flowshop is a function \(\Omega : O \rightarrow \mathbb{R}_{\geq 0}\), which describes the starting times for all operations in \(O\). The schedule is called feasible if the following constraints for all operations in \(O\) are met:

1) Machines must be visited in the fixed sequence defined by the sequence vector \(\theta\).

2) A machine can execute at most one operation at a time: \(\Omega(o_1) + P(o_1) \leq \Omega(o_2)\) or \(\Omega(o_2) + P(o_2) \leq \Omega(o_1)\) for all \(o_1, o_2 \in O\) for which \(o_1 \neq o_2\).

3) A consecutively scheduled operation on a machine can only start after the previous operation is completed and the machine is set up. Thus for any consecutive scheduled operations \(o_1, o_2 \in O\) on the same machine it holds that \(\Omega(o_2) \geq \Omega(o_1) + P(o_1) + S(o_1, o_2)\).

4) If a due date \(D(o_{j,a}, o_{j,a+1})\) is defined, then after \(o_{j,a}\) has started, \(o_{j,a+1}\) can start no later than the relative due date of \(o_{j,a}: \Omega(o_{j,a+1}) \leq \Omega(o_{j,a}) + D(o_{j,a}, o_{j,a+1})\). Due dates are defined between consecutive operations of the same job.

5) The last operation of a job \(j \in J\) is executed before the last operation of job \(j+1\); i.e. \(\Omega(o_{j,t(j)}) < \Omega(o_{j+1,t((j+1))})\). Thus sheets leaving the last machine (POM) are completed in order, specified by \(j\).

6) A job cannot overtake other jobs in-between machines. For any jobs \(s, t \in J\) follow the same route and \(s\) enters the LSP before \(t\), then \(s\) cannot arrive at any machine earlier than \(s\): for all \(s, t \in J\) with \(r(s) = r(t)\) and \(\Omega(o_{s,1}) < \Omega(o_{t,1})\): for all \(1 \leq u \leq r(s): \Omega(o_{s,u}) < \Omega(o_{t,u})\).

Defintion 3. (Schedule quality) The quality of a schedule \(\Omega\) is measured by its makespan \(MS\), which is the maximum of completion times of operations in \(O\). Since sheets at the output are in a fixed order, the makespan is the completion time of unloading the last scheduled sheet of a print job. I.e. \(MS = \Omega(o_{b,r(j_k)}) + P(o_{b,r(j_k)})\) with \(j_k\) the last job in \(J = \{j_1, \ldots, j_k\}\).

The scheduling objective is to find feasible schedules with minimal makespan.

IV. SCHEDULING MODEL

We use a constraint graph to model the constraints of the re-entrant flowshop problem. A constraint graph represents the constraints of a problem with vertices that are connected by weighted arrows. [1] defines the constraint graph model that we use in this paper. We create an ordering of operations to achieve a total order on each machine by encoding constraints of ordering decisions in the graph. Furthermore, we check feasibility of ordering decisions and derive the earliest feasible starting times of operations in the re-entrant flowshop problem from the constraint graph.

A. Constraint graph

Fig. 2 shows a constraint graph model of a print job with five simplex jobs. By C5 of Definition 2, the order of unloading jobs at the POM (the last operation) is fixed. At every segment, all jobs with the same route obey the order in which they are unloaded by the LSP (by C6 of Definition 2). Therefore the operations in the ITM must also obey the order of unloading jobs. With similar reasoning, operations in the PIM must also obey the order of unloading jobs at the POM. In the constraint graph model of Fig. 2 the horizontal black edges enforce these
constraints between all operations of the same machine pass; i.e. vertices in the same row. The sequence of visited machines by jobs, constraint C1 of Definition 2, is enforced by the vertical black edges. The red dashed edges indicate the defined due dates of constraint C4.

B. Total ordering

The simplex jobs of the scheduling problem in Fig. 2 have no re-entrancy. The total order on all three machines is enforced by: (1) the order of unloading jobs and (2) all sheets follow the simplex route (C5 and C6 of Definition 2). Therefore there is no conflict between operations on the same machine. A simplex flowshop problem is trivial. There is no scheduling freedom and it is totally ordered. It merely requires computing the operation starting times of operations.

Fig. 3 represents the constraint graph model of a flowshop for a duplex print job. Jobs in this scheduling problem visit the second machine twice. Operations of the PIM and POM are totally ordered by following the order of unloading. In the ITM, all first passes (upper row) must obey the order in the POM machine. The same applies to all second passes (lower row). The scheduling freedom of this flowshop problem is the order of operations on the re-entrant machine (ITM). The scheduler can choose between feeding a new sheet or a re-entrant sheets to join the stream of sheets for the ITS by determining the order at the Merge Point of the LSP.

The total ordering and the associated setup times are enforced by adding edges with sequence dependent setup times between the first and second passes (green edges in Fig. 3). Without these edges, the first and second pass operations may violate constraints C2 and C3 of Definition 2. For example, the green edge between $o_{3,2}$ and $o_{2,3}$ of Fig. 3 enforces the selected order. With the green edges all operations on the ITM machine are totally ordered.

The well-known Bellman-Ford algorithm (BFA) [15], [16] is used to check feasibility of a constraint graph model. It computes the longest paths from the first duplex PIM operation to all other operations. If the algorithm converges, the constraint graph is feasible, and has determined the earliest possible starting times of operations that make up a feasible schedule.

C. Mixed-plexity scheduling model

When jobs of both duplex and simplex types are used in a single print job, the scheduling freedom changes. The job completion order still determines the order of operations in the POM (i.e. last operation of a job). Since simplex and duplex jobs follow a different route, the order of unloading does not necessarily enforce a fixed order of operations on the earlier machines, in contrast to the jobs in Fig. 2 and 3. The segments of the LSP are visualized for both job types in Fig. 4 and shows their relation to edges in a constraint graph model.

Jobs cannot overtake each other on a segment (by constraint C6 of Definition 2). The vertical edges in the graph correspond to travelling along a certain segment. We note that for a simplex job the first and last ITM operation are the same. Thus all jobs are printed by the ITM in the order in which they enter the PIM, as they share the purple colored segment. The orange re-entrant segment is only taken by duplex jobs for the second pass. Simplex jobs directly take the green segment towards the POM for unloading. Thus there is no direct relationship between simplex sheets and the first passes of duplex sheets.

The other part of the track that both job types share is from the end of the last ITM operation until the start of the POM operation (colored green). Jobs must be unloaded in a fixed order, which directly limits the possible order of the last ITM operations of both job types.

For mixed-plexity problems, in addition to determining the order in which jobs re-enter the ITM, the scheduler must determine the order in which jobs enter the LSP. Jobs in mixed-plexity scheduling can enter the LSP in a different way than they are unloaded. This extra scheduling freedom in the order of the PIM is referred to as out-of-order separation. The mixed-plexity scheduling problem requires scheduling decisions to achieve a total ordering for two machines.
Fig. 4: Segments of the LSP highlighted in the paper path and a 4-job mixed-plexity constraint graph model. Purple segment is between the PIM and ITM, the green segments is between the ITM and POM. The re-entrant segment is highlighted orange.

An example of the benefit of out-of-order separation compared to in-order separation is illustrated in Fig. 5. It shows the processing a print job when ordering them in-order and out-of-order. We consider a mixed-plexity print job of four sheets with constraints as shown in the 4-job constraint graph of Fig. 4. With in-order separation the ITM is idle when duplex sheets travel in the orange colored re-entrant path. The travel time of sheets in this segment results in significant idle time. With out-of-order scheduling the duplex jobs enter the LSP first such that idle time by travelling the re-entrant path is minimized. The simplex job $s_2$ arrives at the ITM before the second pass of $s_3$ is processed. In this example, out-of-order separation achieves 31% better makespan than in-order scheduling. The re-entrant path that is taken by duplex jobs allows scheduling simplex jobs in between. In this way, jobs are separated out-of-order and are unloaded in-order.

V. SCHEDULING HEURISTIC

We first present the scheduling outline before investigating the heuristic in detail. The scheduling outline of HCS [1] is the basis for the MPHCS algorithm. MPHCS follows the structure of HCS, but adds functionality to schedule variable re-entrant jobs. The scheduling mechanism for fixed re-entrant jobs stays the same. The scheduling approach of (MP)HCS is to add one job to the sequence in each iteration, until a complete feasible schedule is obtained. Per iteration MPHCS executes four steps:

1. Find all potentially feasible schedule options for the job of the current iteration.
2. Estimate and rank the quality of the schedule options.
3. Checking feasibility of the best ranked schedule option until a feasible schedule option is found.
4. Add constraints of the schedule option to the constraint graph.

A. Ordering method

In each iteration a job is ordered in an existing constraint graph. To achieve a total order for the ITM machine, second passes of duplex jobs and ITM operations of simplex jobs are ordered between the first passes of duplex jobs. I.e., ITM operations in the lower row in Fig. 4 are ordered between operations in the upper row. Lower passes of duplex jobs therefore form the initial sequence. A total order in the PIM machine is achieved by adding the PIM operation of simplex jobs in each iteration to the existing sequence. Its initial sequence consists of all PIM operations of duplex jobs.

B. Heuristic

The MPHCS is given in Algorithm 1.

1) **Initial constraint graph:** Initially, a constraint graph is created from the scheduled flowshop by `CREATE_INITIAL_CONSTRAINT_GRAPH(f)`. The constraint graph contains one vertex for each operation and constraints are encoded by edges [1]. The vertices are constrained by applying the scheduling constraints on the unordered set of operations as illustrated in Fig. 4. After creating the constraint graph, an initial lower bound on each of the start times of the operations is computed by `DETERMINE_INITIAL_START_TIMES(cg)`.

2) **Schedule options:** In each iteration the heuristic selects a set of potential feasible schedule options for the scheduled job by the function `CREATE_SCHEDULE_OPTIONS(f, j)`. A schedule option is defined as an ordering tuple of fixed re-entrant jobs stays the same. The scheduling approach of (MP)HCS is to add one job to the sequence in each iteration, until a complete feasible schedule is obtained. Per iteration MPHCS executes four steps:

   1. Find all potentially feasible schedule options for the job of the current iteration.
   2. Estimate and rank the quality of the schedule options.
   3. Checking feasibility of the best ranked schedule option until a feasible schedule option is found.
   4. Add constraints of the schedule option to the constraint graph.

Fig. 5: Example of the benefit of out-of-order separation. It shows the duration of segments when separating 4-sheet print job in-order and out-of-order. The print job alternates with duplex and simplex sheets.

<table>
<thead>
<tr>
<th>In-order</th>
<th>Out-of-order</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$(duplex)</td>
<td>$s_1$(duplex)</td>
</tr>
<tr>
<td>$s_2$(simplex)</td>
<td>$s_2$(simplex)</td>
</tr>
<tr>
<td>$s_3$(duplex)</td>
<td>$s_3$(duplex)</td>
</tr>
<tr>
<td>$s_4$(simplex)</td>
<td>$s_4$(simplex)</td>
</tr>
</tbody>
</table>

4) Add constraints of the schedule option to the constraint graph.
it requires two ordering tuples. Fig. 6b shows an example ordering tuples to schedule a simplex job.

There is a direct relation in the schedule option of the PIM and ITM operation for simplex jobs: the order that is selected order between $a$, $b$ and $c$ and the due dates of operations after the set is scheduled out-of-order. By choosing schedule options that enable out-of-order separation in the PIM machine, the thickened order in the ITM is enforced. The jobs are separated out-of-order and unloaded in-order.

The proposed scheduling method with ordering tuples provides a powerful tool to create schedules that enable out-of-order separation for mixed-plexity. Fig. 7 shows an example with the same job set as in Fig. 6b. In this example, the job set is scheduled out-of-order. By choosing schedule options that enable out-of-order separation in the PIM machine, the thickened order in the ITM is enforced. The jobs are separated out-of-order and unloaded in-order.

The set of schedule options includes all options that meet the ordering constraints. For example, the second pass of a duplex job must be scheduled after its first pass. Feasibility of a schedule option is still unknown, because it might not be possible to find starting times that satisfy the sequence dependent setup times and the due dates of operations after inserting the schedule option. This is verified later, in Step 4.

3) Estimate and rank quality: The quality of a schedule is quantified by its makespan. The makespan depends on the chosen schedule option; (1) directly impacting the makespan by the chosen sequence, and (2) indirectly influencing the makespan by limiting the potential feasible set of schedule options in next iterations. Two metrics, productivity and flexibility, are used to estimate the quality of scheduling options.

The productivity metric quantifies the direct impact of a schedule option on the makespan of a schedule. It measures the maximum increase in start time for operation $c$ by inserting the ordering tuple $ot((a, b))(b, c)$ of a schedule option. An increase in begin time for vertex $c$ propagates the start time of all consecutive scheduled jobs. $d_b$ computes the maximum increase in start time for $c$ that results from any ordering tuple $ot((a, b))(b, c)$ in the set of potentially feasible schedule options $OT_j$ of a scheduled job $j$.

$$d_b = \max_{\omega \in OT_j} \max(\Omega(c), \max(\Omega(b), \Omega(a) + w(a, b)) + w(b, c))$$ (1)

A low productive ordering tuple causes a large increase in makespan. $P_{ot} = 0$ indicates that the ordering tuple does not directly increase the makespan and is maximally productive. Productivity is only measured for the ITM machine. In the considered LSP scheduling problem the productivity is not affected by schedule decisions in the PIM machine. In case
the input machine does matter, the formula can be updated to take into account the productivity of the PIM machine as well.

\[ P_{ot} = \max(\Omega(c), \max(\Omega(b), \Omega(a) + w(a,b) + w(b,c)) - \Omega(c)) \]  
\[ \frac{db}{dt} \]

Flexibility indicates the effect of the schedule option of a scheduled duplex job on the remaining slack time of \( b \) (the slack in the re-entrant path buffer). A schedule option can propagate the start time in \( b \). An increase in start time of \( b \) decreases the set of potentially feasible schedule options of the following scheduled jobs. The more slack is left for the due date constraint of \( b \), the more potentially feasible schedule options are left for following scheduled jobs, possibly improving the makespan. The increase in start time of \( b \) is computed in the numerator, while the denominator computes the slack in the re-entrant path buffer). A schedule option can propagate the start time in \( b \) too much. Simplex jobs do not travel the re-entrant path, thus are maximally flexible. For this reason, flexibility is only computed when scheduling a duplex job.

\[ F_{ot} = \max(\Omega(b), \Omega(a) + w(a,b)) - \Omega(b) - w(b, b-1) - w(b-1, b) \]  
\[ \frac{db}{dt} \]

The schedule options are ranked by \texttt{RANK\_SCHEDULE\_OPTION(ot)} as determined by Equation 4. The relative weights \( \kappa_p \) and \( \kappa_f \) sum up to 1 and indicate the relative importance of productivity and flexibility. The schedule option with minimum rank is picked.

\[ R_{ot} = \kappa_p \times P_{ot} + \kappa_f \times F_{ot} \]  
\[ \frac{db}{dt} \]

4) \texttt{Check feasibility and compute start times:} The function \texttt{CHECK\_FEASIBILITY\_CHOICE(ot, cg)} checks the feasibility of the minimum ranked schedule option in \( OT_j \). The BFA [1], [2] is used to check feasibility and compute starting times after inserting the schedule option constraints. If it detects a positive cycle in the constraint graph, then the schedule with the inserted schedule option is not feasible. When the schedule is not feasible, the next schedule option is picked. This is repeated until a feasible schedule option is found. The constraints of the feasible schedule option are inserted into the constraint graph and the next job is scheduled. In addition to checking feasibility, the BFA computes the starting times of operations, providing the schedule \( \Omega \).

C. Worst-case complexity

The worst-case complexity of BFA for this scheduling problem is \( O(|V| \times |E|) = O(|J|^2 \times r^2) \), because the number of edges \( |E| \) are related to the number of vertices \( V \) in this scheduling problem. The heuristic evaluates at most \( |J|(|J|+1)/2 \) schedule options, thus the worst-case complexity to produce a schedule with this heuristic is \( O(|J|^4 \times r^2) \).

VI. Experimental evaluation

The experiments aim to assess the scheduling quality of the MPHCS by comparing the makespan of produced schedules with estimated lower bounds of the optimal makespan, using a translation of the flowshop problem to Mixed Integer Programming (MIP). We describe the experimental setup and compare the makespan in this section.

A. Experimental setup

Experiments have been executed on a 3.0GHz Intel i7 processor with 8GB RAM. All algorithms are implemented single threaded. The MIP translations have been run on CPLEX 12.7 and is run at most \( |J| \times 1 \) seconds per schedule request \( J \).

B. Test set

The MPHCS and CPLEX model are compared by set of 1719 typical schedule requests for an LSP. 1399 requests are mixed-plexity job sets and 320 job sets consist of only simplex jobs. Job sets vary in size between 30 and 810 jobs. In addition to plexity type, sheets can also vary in size, thickness, media type and required print height. Table I presents the mixed-plexity type, and is run at most \( |J| \times 1 \) seconds per schedule request \( J \).

\|
<table>
<thead>
<tr>
<th>Category</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>( S^+D^+ )</td>
</tr>
<tr>
<td>VAR</td>
<td>( ((S^n(S))(S))(S)(D)(S))^{1+} )</td>
</tr>
<tr>
<td>T</td>
<td>( S^+D^+ )</td>
</tr>
<tr>
<td>RA</td>
<td>( (S(D))^+</td>
</tr>
<tr>
<td>RB</td>
<td>( ((S</td>
</tr>
<tr>
<td>TAB</td>
<td>( (S(D)^n)^{10} )</td>
</tr>
<tr>
<td></td>
<td>( n \in {50, 60, 70, 80} )</td>
</tr>
<tr>
<td>Simplex</td>
<td></td>
</tr>
</tbody>
</table>

TABLE I: Patterns in the mixed-plexity and simplex job sets.

\|
<table>
<thead>
<tr>
<th>Category</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>( a^+ )</td>
</tr>
<tr>
<td>RA</td>
<td>( a^+b^+a^+b^+a^+ )</td>
</tr>
<tr>
<td>RB</td>
<td>( a^{10}b^{10}c^{10}d^{10}e^{10} )</td>
</tr>
</tbody>
</table>

C. Experimental method

Duplex test cases are already thoroughly tested by [1]. The scheduling algorithm for duplex job sets has not changed by extending the HCS, which will result in the same makespans as in [1]. We are therefore more interested in the performance of scheduling mixed-plexity job sets. The simplex patterns are presented as regular expressions with literals \( a-e \), where literals represent a different sheet type.

The MPHCS and CPLEX model are compared by set of 1719 typical schedule requests for an LSP. 1399 requests are mixed-plexity job sets and 320 job sets consist of only simplex jobs. Job sets vary in size between 30 and 810 jobs. In addition to plexity type, sheets can also vary in size, thickness, media type and required print height. Table I presents the mixed-plexity type, and is run at most \( |J| \times 1 \) seconds per schedule request \( J \).

\|
<table>
<thead>
<tr>
<th>Category</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>( S^+D^+ )</td>
</tr>
<tr>
<td>VAR</td>
<td>( ((S^n(S))(S))(S)(D)(S))^{1+} )</td>
</tr>
<tr>
<td>T</td>
<td>( S^+D^+ )</td>
</tr>
<tr>
<td>RA</td>
<td>( (S(D))^+</td>
</tr>
<tr>
<td>RB</td>
<td>( ((S</td>
</tr>
<tr>
<td>TAB</td>
<td>( (S(D)^n)^{10} )</td>
</tr>
<tr>
<td></td>
<td>( n \in {50, 60, 70, 80} )</td>
</tr>
<tr>
<td>Simplex</td>
<td></td>
</tr>
</tbody>
</table>

TABLE I: Patterns in the mixed-plexity and simplex job sets.

\|
<table>
<thead>
<tr>
<th>Category</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>( a^+ )</td>
</tr>
<tr>
<td>RA</td>
<td>( a^+b^+a^+b^+a^+ )</td>
</tr>
<tr>
<td>RB</td>
<td>( a^{10}b^{10}c^{10}d^{10}e^{10} )</td>
</tr>
</tbody>
</table>
create a total order on both machines. This scheduling method used to efficiently extend the HCS scheduling method to shown that the properties of the scheduling problem can be products can enter the FMS into the first machine. We have jobs gives additional scheduling freedom in the order that entrant machine. The variable re-entrant scheduling problem only require ordering decisions of products passing the re-entrant job sets, where jobs does not negatively affect the scheduling quality for fixed re-entrant job sets and shown that the MPHCS algorithm is a good scheduling mechanism for variable re-entrant scheduling.

D. Experimental results

Figure 8 shows the distribution of the percentage of the makespan over the lower bound by the MPHCS algorithm. MPHCS finds schedules with a makespan that lies within 10% of the lower bound for 50% of the test cases. The MIP algorithm has found an optimal schedule for 53 of the 1399 mixed-plexity test cases. The lower bound produced by the MIP algorithm might not be tight. If CPLEX has not found an optimal solution for a given test case in time, then it returns the best lower bound it has found. The deviation of the lower bound from the optimal schedule is unknown. However, the resulting makespan by MPHCS gives an indication. Especially for large test cases with many jobs it has difficulty to find an optimal schedule. This increases the deviation between makespans from MPHCS schedules and the produced lower bound from the optimal schedule is unknown. However, the experimental results confirm that the MPHCS can produce good quality schedules for variable re-entrant job sets. The makespans are in at least 50% of the test cases within 10% of the optimal makespan. We have provided an extension that does not negatively affect the scheduling quality for fixed re-entrant job sets and shown that the MPHCS algorithm is a good scheduling mechanism for variable re-entrant scheduling.

VII. Conclusion

A heuristic is presented for a variable re-entrant scheduling problem and is applied to a 3-machine FMS with fixed order, where products can (re)visit the re-entrant machine once or twice. The MPHCS heuristic extends the HCS [1] heuristic that schedules fixed re-entrant job sets, where jobs only require ordering decisions of products passing the re-entrant machine. The variable re-entrant scheduling problem gives additional scheduling freedom in the order that products can enter the FMS into the first machine. We have shown that the properties of the scheduling problem can be used to efficiently extend the HCS scheduling method to create a total order on both machines. This scheduling method significantly increases the scheduling quality for variable re-entrant job sets, compared to in-order scheduling, by the ability to create schedules where products enter FMS in a different order in which they must leave the FMS.

The experimental results confirm that the MPHCS can produce good quality schedules for variable re-entrant job sets. The makespans are in at least 50% of the test cases within 10% of the optimal makespan. We have provided an extension that does not negatively affect the scheduling quality for fixed re-entrant job sets and shown that the MPHCS algorithm is a good scheduling mechanism for variable re-entrant scheduling.

References