Analysis And Comparison Of Dispatching Rule-Based Scheduling In Dual-Resource Constrained Shop-Floor Scenarios

Bernd Scholz-Reiter, Jens Heger, Torsten Hildebrandt

Abstract—There has been a lot of research about scheduling manufacturing systems with dispatching rules in the past. However, most studies concentrate on simplified scenarios considering only one type of resource, usually machines.

In this study dispatching rules are applied to a more realistic scenario, which is dual constrained by machines and operators and has a re-entrant process flow. Interdependencies of dispatching rules are analyzed by long-term simulation. Strength and weaknesses of various priority rule combinations are determined under different utilization levels. To gain further insights into the problem we additionally solve static instances optimally and with priority rules. For the optimal solutions we extend a mixed integer linear program (MILP) of the production system to cover operators and re-entrant process flows.

Index Terms—Discrete-event simulation, dual-resource constraints, MILP, priority rule scheduling, shop-floor control

I. INTRODUCTION

Shop-Scheduling has attracted researchers for many decades and is still of big interest, because of its practical relevance and the fact that optimal solutions can only be found for very small instances due to its np-complete character. Multi- or dual resource problems are significantly less analyzed, despite being more realistic Scheduling with priority/dispatching rules is quite appealing for various reasons (see section II.B) and often used, especially when considering more realistic and complex manufacturing systems.

This work analyses the quality of dispatching rules in the dual-resource constrained case. Different combinations of rules are tested in order to analyze their interdependencies. Our analysis combines simulation with the optimal solution of static instances. To evaluate the results of the simulation a mixed integer linear program (MILP) has been developed to calculate optimal solutions.

The paper is organized as followed: In chapter 2 a short literature review about shop scheduling, dispatching rules and dual constrained scheduling is given. This is followed by the problem description in chapter 3. In chapter 4 small instances in comparison with optimal solutions are analyzed by the problem description in chapter 3. In chapter 4 small instances optimally and with priority rules. For the optimal solutions we extend a mixed integer linear program (MILP) of the production system to cover operators and re-entrant process flows.

II. LITERATURE REVIEW

In this chapter a literature review is given considering the shop job shop problem, dispatching rules and the dual constrained scheduling.

A. Shop scheduling

Haupt [1] gives a definition of the scheduling or sequencing problem as “the determination of the order in which a set of jobs (tasks) \( \{i=1, ..., n\} \) is to be processed through a set of machines (processors, work stations) \((k=1, ..., m)\)”. Usual abstractions of real-world scheduling problems are the job shop and flow shop problem. With both problem types the number of operations and their sequence are known in advance and fixed. In a job shop scenario each job can have its own route whereas in the classical flow shop all jobs share a common routing. Because of its high complexity (np-hard) and its high practical relevance the problem has attracted researchers and practitioners for decades now. Due to its high complexity optimal solutions can only be calculated for small instances. The 10x10 (10 machines, 10 jobs) Fisher Thompson [2] model for example remained unsolved for over two decades.

To still find good solutions for such problems many heuristics, which do not guarantee to find an optimum, are proposed. Examples are:

- Shifting bottleneck [3]
- Simulated annealing [4]
- Taboo search [5]
- Genetic algorithms (e.g. [6] [7]).

Many of these approaches however focus on quite restrictive assumptions of their scenarios. Extending these often quite complex heuristics to more realistic is usually not straightforward. New, flexible approaches and different solutions need to be applied. Decentralized planning and scheduling as well as smart autonomous items in our opinion are a very promising approach for this (e.g. [8]). Decentralized decisions can be based on local decision rules. This is where dispatching rules come into play.

B. Dispatching rules

Dispatching rules are applied to assign a job to a resource (machine/operator/etc.). This is done each time the resource gets idle and there are jobs waiting or a new job arrives at an idle resource. The dispatching rule assigns a priority to each job. This priority can be based on attributes of the job, the resource or the system. The job with the highest priority is chosen to be processed next.

Priority-scheduling rules have been developed and analyzed for many years [1] [9] [10] [11] [12]. They are...
widely used in industry, especially in complex manufacturing systems, e.g. semiconductor manufacturing. Their popularity is derived from the fact that they perform reasonably well in a wide range of environments, are relatively easy to understand, because of their intuitive nature. They also need only minimal computational time, which allows them to be used even in real-time environments.

Depending on the manufacturing system and the various objectives (mean flow time, maximum flow time, variance of flow time, proportion of tardy jobs, mean tardiness, maximum tardiness, variance of tardiness, etc.) no single rule has been found, which outperforms all others. [10] As a result there are also approaches to find new dispatching rules or choose appropriate ones according to the system’s current state. (e.g. [11] [12] [13])

C. Multiple / dual resources constrained scheduling

Most research has been done on the machine-only constrained problem, where machines are the only limiting resource. Nevertheless, closer to the ‘real’ world are multi or dual constrained (DRC) problems, where more than one resource restricts the output of the system and impacts the shop performance. Gargeya and Deane [14] defined the multiple resource constrained job shop as “a job shop in which two or more resources are constraining output. The resources may include machines, labor and auxiliary resources. Dual constrained job shops are constrained by two resources (machine and labor, machines and auxiliary resources or labor and auxiliary resources). Dual constrained job shops are thus a specific type of multiple resource constrained job shops.”

To solve the multi-resource constrained problem different approaches were proposed. Mati and Xie [15] developed a greedy heuristic. This heuristic is guided by a genetic algorithm in order to identify effective job sequences. Dauzère-Pérès et al. [16] [17] developed a disjunctive graph representation of the multi-resource problem and proposed a connected neighborhood structure, which can be used to apply a local search algorithm such as taboo search. Patel et al. [18] proposed a genetic algorithm for dual resource constrained manufacturing and they compared different dispatching rules against different performance measures.

In the study of Chen et al. [19], an integer optimization formulation with a separable structure is developed where both machines and operators are modeled as resources with finite capacities. By relaxing resource capacity constraints and portions of precedence constraints, the problem is decomposed into smaller sub-problems that are effectively solved by using a dynamic programming procedure. The multipliers are updated using the surrogate sub gradient method. A heuristic is then used to obtain a feasible schedule based on sub-problem solutions.

III. PROBLEM DESCRIPTION

ElMaraghy et al. [20] defined the machine / worker / job scheduling problem as: “Given process plans for each part, a shop capacity constrained by machines and workers, where the number of workers is less than the number of machines in the system, and workers are capable of operating more than one machine, the objective is to find a feasible schedule for a set of job orders such that a given performance criteria is optimized.” An interesting object of investigation in DRC is the interaction effect of the resource constraints and how they impact the performance measure like the mean flow time.

The experimental design used in this study corresponds to the MiniFab scenario [21] [22], which is shown schematically in Fig. 1. For practical reasons the model has been simplified, nevertheless the main properties remain.

The MiniFab scenario is a simplification of a semiconductor manufacturing system, but still containing the characteristics of such a system. It consists of 5 machines with 6-step re-entrant processes. Operators are used to load and unload machines. Machines $Ma$ and $Mb$ as well as $Mc$ and $Md$ are parallel identical machines sharing a common buffer. To be able to get optimal solutions we do not consider sequence-dependent setup times, parallel batching and machine break downs at the moment.

Operations at the machines are divided into three parts: loading, processing and unloading. Processing starts directly after the machines are loaded. Unloading begins after processing, but if there is no operator available for unloading, the machine stays idle. In table 1 processing, loading and unloading times are listed, which are used for the calculations and simulations in this paper.

Table 1: Processing and (un)loading times used

<table>
<thead>
<tr>
<th></th>
<th>$Ma$</th>
<th>$Mb$</th>
<th>$Mc$</th>
<th>$Md$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step1</td>
<td>Ma / Mb</td>
<td>15</td>
<td>55</td>
<td>25</td>
</tr>
<tr>
<td>Step2</td>
<td>Me / Md</td>
<td>20</td>
<td>25</td>
<td>15</td>
</tr>
<tr>
<td>Step3</td>
<td>Me</td>
<td>15</td>
<td>45</td>
<td>15</td>
</tr>
<tr>
<td>Step4</td>
<td>Ma / Mb</td>
<td>20</td>
<td>35</td>
<td>25</td>
</tr>
<tr>
<td>Step5</td>
<td>Mc / Md</td>
<td>15</td>
<td>65</td>
<td>25</td>
</tr>
<tr>
<td>Step6</td>
<td>Me</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

The processing and (un)loading times as well as the machine and sequence settings are fixed. The machines select a job out of the waiting queue and call an operator to load them (point in time a in Fig. 2). If no operator is available the machines wait until eventually they are loaded (b-c) and proceed with processing afterwards (c-d).
The four resulting scenarios are:

- **NO_OPERATOR**: we don’t consider operators at all in this case, operations only require machines.
  (5 machines, 0 operator)

- **MACHINE_CONSTRAINED**: three operators are available, making machines the more critical resource.
  (5 machines, 3 operator)

- **OPERATOR_CONSTRAINED**: there is just one operator; therefore operators are much more critical than machines.
  In fact in this scenario the operator becomes the capacity-constraining factor, reducing the maximum long-term system capacity to 6.857 instead of 13.714 orders per day.
  (5 machines, 1 operator)

- **DUAL_CONSTRAINED**: there are two operators in the model, resulting in machines being roughly as critical as operators.
  (5 machines, 2 operator)

We consider FIFO (First In buffer First Out), FSFO (First in System first Out), Rnd (Random) and SPT (Shortest Processing Time first) as sequencing rules for the machines. Besides FIFO, FSFO, Rnd and SPT we consider two additional decision rules for operators: MQL (longest Machine Queue Length first) and SSPT (Shortest Step Processing Time first). With MQL operators give priority to machines with a long queue of waiting jobs. SSPT is similar to SPT, but instead of considering the processing time of the complete operation, only the length of the required load or unload processing steps are used. Furthermore we vary the tie-breaker rule which is used as a secondary criterion if priorities of the primary rule are equal. The 72 combinations of sequencing rules investigated are listed in table 2. Using FIFO, FSFO and Rnd no equal priorities can occur, so there is no need for a tie-breaker rule in these cases.

To decide which job to process next, we use a two-step procedure, first using the machine rule to select a job out of the waiting jobs in front of a machine. This sends a processing request to the operator pool. The operator rule is then used to decide between the requests of the different machines. If no operator is available immediately, the machine is kept in a waiting state.

In this study we only consider the common performance measure mean flow time. Our experiments are divided into two parts. First we have experiments with only a few jobs, which enable us to compare optimal schedules with the schedules the priority rules provide (chapter 4). And secondly we perform a long term simulation study, with a time span of ten years to analyse the priority rules in detail (chapter V).

### IV. EXPERIMENTS WITH STATIC INSTANCES: SOLVER VS. PRIORITY RULES

To determine not only the differences between priority rules and their combinations, it is interesting to analyse their performance in comparison to optimal solutions to see the total deviation and evaluate priority rules in general.

#### A. Experimental design

We simulated instances from the same scenario with only 2-50 jobs in the system. All jobs were released at the same time and known in advance. This simplification makes it possible to calculate optimal solutions for the instances with only a few jobs in the system. For larger instances feasible schedules (solutions of the MILP) could be found, but they were not proven optimal. Due to the complexity of the model, there still was a gap between the found solution (upper bound) and the theoretical possible solution (lower bound).

The optimum results are calculated with CPLEX [23] solving a MILP-formulation, which is an extension of the advanced job shop formulation used by Pan and Chen [24]. Operators and re-entrant processes were added to their flexible job shop MILP, so that the MiniFab model as described above can be solved and optimal schedules are calculated. (A detailed description would extend the scope of this paper and will be published additionally)

In our study, we were able to calculate some optimal solutions, which were in most cases only a bit better than the best rule combination. In table 2 we listed the solver results with the remaining gap and the performance of the best and worst rule combination taken from any of the 72 combinations (see chapter 3). For a detailed analysis of the static runs, we chose the FSFO ShortestStepLength [FSFO] rule, which seems to be the best performing one. In Fig. 3 the corresponding graphs are plotted. The FSFO ShortestStepLength [FSFO] rule is indicated by the green line. The grey bar defines the area between best and worst solver solutions. Black dots correspond with the solver results, gaps between found solutions and lower bounds are printed as lines.

#### B. Analyses of static instances

There need to be about 4 or 5 jobs in the system and first differences between good and bad rule combinations can be found. An example is the DUAL_CONSTRAINED case with 5 jobs in the system: The rule combination FSFO (machine) ShortestStepLength [FSFO] (operator) has a mean flow time of 656 minutes and the combination FIFO (machine) FSFO (operator) has 724 minutes. This is already a difference of more than 10%. The difference to the solver solution is only around 2%.

The results of the scenario with NO_OPERATORS and the MACHINE_RESTRICTED scenario are very similar as...
Fig. 3 and table 2 show. When there are more jobs in the system the performance differences between the rule combinations arises.

In the DUAL_CONSTRAINED case, it can be detected that the performance differences between rule combinations are much higher, and they arise in smaller scenarios. This result was expected, because in the dual constrained case the interdependences between the machine priority rule and the operator priority rule are the strongest and amplify each other.

In the OPERATOR_CONSTRAINED case the shop is restricted the most, which leads very quickly to high mean flow times. The differences between the rule combinations are also very high in this case. The solution the solver provided for the cases with 3 and 4 jobs indicate that either the rule performance or the order of selection (machines first, operators second) do not perform very well in this scenario. It seems more likely that the order of selection is responsible for this effect. The operator is clearly the bottleneck and the scheduling should be arranged in that way that he is utilized as best as possible, which is clearly not the case.

For instances with more jobs we were not able to proof the same effect, because the solver provided no optimal solutions even after days of calculations. The found solutions were comparable to the used rules.

<table>
<thead>
<tr>
<th>number of jobs</th>
<th>2 3 4 5 6 7 8 9 10 20 50</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean flow time in minutes</td>
<td>483 520 566 622 678 733 785 870 918 1424 2989</td>
</tr>
<tr>
<td>gap (lower/upper bound)</td>
<td>0% 0% 0% 0% 0% 8% 0% 27% 37%</td>
</tr>
<tr>
<td>best rule</td>
<td>483 520 566 628 684 739 897 918 1437 3042</td>
</tr>
<tr>
<td>worst rule</td>
<td>483 520 566 637 716 804 905 967 1040 1940 4616</td>
</tr>
<tr>
<td>span best-worst</td>
<td>0% 0% 0% 1% 5% 7% 10% 11% 15% 36% 54%</td>
</tr>
<tr>
<td>FSFO SSPT [FSFO]</td>
<td>483 520 566 628 684 739 897 918 1437 3042</td>
</tr>
<tr>
<td>MACH_CONSTRAINED</td>
<td>optimum 483 520 566 622 gap (lower/upper bound) 0% 0% 0% 0% best rule 483 520 566 628 671 716 804 905 967 1040 1940 worst rule 483 520 566 625 675 716 804 905 967 1040 1940 span best-worst 0% 0% 0% 1% 5% 7% 10% 11% 15% 36% 54% FSFO SSPT [FSFO] 483 520 566 628 684 739 897 918 1437 3042</td>
</tr>
<tr>
<td>DUAL_CONSTRAINED</td>
<td>optimum 483 520 566 622 gap (lower/upper bound) 0% 1% 0% 3% 9% 19% best rule 483 520 569 628 684 739 897 918 1437 3042 worst rule 483 520 569 628 684 739 897 918 1437 3042 span best-worst 0% 1% 5% 11% 16% 17% 18% 19% 21% 34% 45% 66% FSFO SSPT [FSFO] 483 520 569 628 684 739 897 918 1437 3042</td>
</tr>
<tr>
<td>OP_CONSTRAINED</td>
<td>optimum 483 520 566 622 gap (lower/upper bound) 0% 0% 0% 33% 38% 47% best rule 535 643 756 917 1013 1118 1225 1331 1486 2541 5695 worst rule 548 685 842 1041 1243 1423 1616 1807 2022 4224 10501 span best-worst 2% 6% 11% 23% 33% 64% 105% 139% 258% 500% 1382% FSFO SSPT [FSFO] 546 643 803 918 1033 1148 1253 1358 1491 2641 5695</td>
</tr>
</tbody>
</table>

Figure 3: Results for all static cases. Green line: FSFO ShortestOpStepLength [FSFO] rule, grey area: best/worst rule combination result, black points: Solver results with gap, if any (line)
V. LONG-TERM SIMULATION

To validate the results from our static analyses we perform an extensive simulation study with a time horizon of ten years.

A. Long-term simulation

To assess the performance of the various rule combinations in the long term we simulate the system under three load conditions for each of the 4 scenarios: 70%, 80% and 90% bottleneck utilization. Given the scenario and load level we determine the appropriate arrival rate based on a static analysis of system capacity. Interarrival times follow an exponential distribution.

We simulate a time span of 10 years, ignoring data from the first year in our results. Altogether we determine system performance for 864 different parameter settings (4 scenarios with 3 load levels each, 72 rule combinations). For the different parameter settings we use common random numbers as a variance reduction technique [25]. The results presented here are the averages of the mean flow time achieved in 20 replications.

B. Analysis of long term simulations

Fig. 3 shows graphically the scenario and load level setting and the best rule result achieved in this case. The four lines in the xy-plane correspond to our scenarios:

- NO_OPERATOR – green line
- MACHINE_CONSTRAINED – blue line
- DUAL_CONSTRAINED – magenta line
- OPERATOR_CONSTRAINED – dark blue line.

Selecting one of the scenarios fixes the achievable combination of machine and operator utilization along the respective line. To give an example: if MACHINE_CONSTRAINED (blue) is chosen and a load level of 90% bottleneck utilization (in this case: machine utilization), this results in an operator utilization of 60%. The grey bar at this point shows the best flow time achieved, i.e. out of the 72 rule combinations investigated.

The best results can obviously be achieved if no operator constraints are present at all. This corresponds to the green line (NO_OPERATOR) in our simulation; we also have the lowest flow times in this scenario, compared to the other scenarios at the same load level. The DUAL_CONSTRAINED case (magenta) on the other hand is the most difficult, operator and machines are equally critical. This can also be seen in Fig. 2.

In summary, all four scenarios show that the performance of dispatching rules differs in some cases tremendously. The interdependences of the rules especially in the DUAL_CONSTRAINED scenarios lead to high performance differences. Other system factors, e.g. the utilization rate also affect the results.
Table 3: Results for all 72 rule combinations investigated. Except for the last row (span) all values are flow factors, i.e. mean flow time divided by the processing time (445 minutes). The last column contains a mean performance averaged over all scenario level combinations (but excluding DUAL CONSTRAINED / 90%, see text); so (system overflow) more incoming than outgoing jobs.

<table>
<thead>
<tr>
<th>machine rule</th>
<th>operator rule</th>
<th>NO_OPERATOR (0 Operator)</th>
<th>MACH_CONSTRAINED (1 Operator)</th>
<th>OPR_CONSTRAINED (2 Operator)</th>
<th>DUAL_CONSTRAINED (3 Operator)</th>
<th>Mean (flow factor)</th>
<th>utilisation of bottleneck</th>
<th>utilisation of bottleneck</th>
<th>utilisation of bottleneck</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIFO</td>
<td>FIFO</td>
<td>1.32 1.52 2.13</td>
<td>1.34 1.58 2.32</td>
<td>1.49 1.44 2.67</td>
<td>3.26 2.65 3.60</td>
<td>2.22 2.68</td>
<td>80% 90% 90%</td>
<td>80% 90% 90%</td>
<td>80% 90% 90%</td>
</tr>
<tr>
<td>FIFO</td>
<td>FIFO</td>
<td>1.32 1.52 2.13</td>
<td>1.34 1.58 2.32</td>
<td>1.49 1.44 2.67</td>
<td>3.26 2.65 3.60</td>
<td>2.22 2.68</td>
<td>80% 90% 90%</td>
<td>80% 90% 90%</td>
<td>80% 90% 90%</td>
</tr>
</tbody>
</table>

WCECS 2009
Proceedings of the World Congress on Engineering and Computer Science 2009 Vol II
WCECS 2009, October 20-22, 2009, San Francisco, USA

WCECS 2009
VI. CONCLUSION AND FURTHER RESEARCH

The paper analyses a priority-rule based approach to shop-floor scheduling considering both machines and operators. Our dual-constrained scenario derived from semiconductor manufacturing. To gain insights into priority rule performance we use two approaches: simulation of small problem instances and its comparison with optimal solutions and simulation of long term system behavior. Especially in shop environments, where more than one resource is constraining the system’s performance proper selection and combination of priority rules is crucial. The long term simulation has also shown, that in high utilization scenarios many rule combinations are not able to handle the workload at all, which underlines the importance of choosing the right combination.

Comparison of known optimal solutions and best rule results indicates only a small difference between the optimum and the best rule result. The differences between well performing rule combinations and unsuitable combinations is much higher than the differences to optimal solutions, which shows that well adjusted priority rules provide good solutions. It would be interesting to see if this remains true for larger problem instances, so a more efficient optimal solution procedure is needed here.

It is interesting to see, that the rule combination’s performance depends on the utilization level of the system. This brings up the idea to develop a control system, which selects or develops priority rules automatically, considering system parameters and adopting to changing shop floor conditions.

REFERENCES


