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

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 and in chapter 5 a long term simulation study is described. The paper closes with a conclusion and description of future research.

Manuscript received July 26, 2009. This work was supported by the German Research Foundation (Deutsche Forschungsgemeinschaft DFG).

Bernd Scholz-Reiter, Jens Heger and Torsten Hildebrandt are with the Bremer Institut für Produktion und Logistik (BIBA) at the University of Bremen, Hochschulring 20, 28359 Bremen, Germany (corresponding author: Jens Heger, +49 421 218 9788; heg@biba.uni-bremen.de)

II. Literature review

In this chapter a literature review is given considering the 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|i = 1, ..., n) is to be processed through a set of machines (processors, work stations) (k|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.

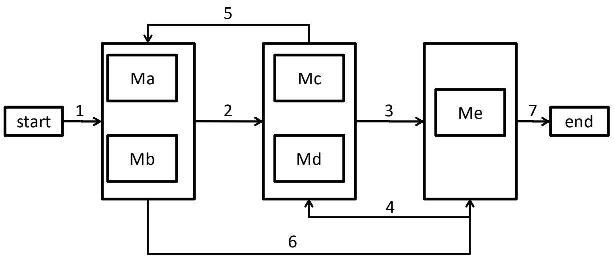

Figure 1 : Process flow of the 5 Machine 6 step production model

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

	machine	**loading**	**processing**	**unloading**
Step1	Ma / Mb	15	55	25
Step2	Mc / Md	20	25	15
Step3	Me	15	45	15
Step4	Ma / Mb	20	35	25
Step5	Mc / Md	15	65	25
Step6	Me	10	10	10

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).

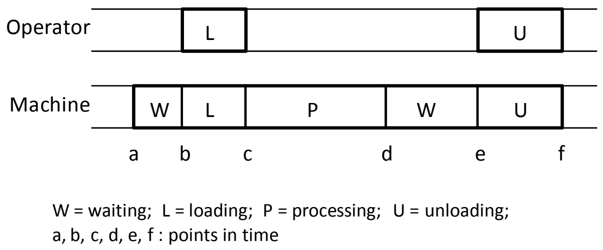
Operator
L
U
Machine
W
L
P
W
U
a
b
c
d
e
f
W = waiting; L = loading; P = processing; U = unloading;
a, b, c, d, e, f : points in time

Figure 2: Gantt chart illustration of machine-operator assignment

If an operator is available, unloading is performed immediately (e-f), otherwise there is an additional waiting period (d-e). The operators are not needed during processing and can work on other machines in that time period.

We vary the number of operators from 0 to 3 to create scenarios constraining the available capacity in different ways. 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 ShortestOpStepLength [FSFO] rule, which seems to be the best performing one. In Fig. 3 the corresponding graphs are plotted. The FSFO ShortestOpStepLength [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) ShortestOpStepLength [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 2: Results of static scenarios in minutes

[mean flow time in minutes]	number of jobs										
	2	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**20**	**50**
NO_OPERATOR											
optimum	483	520	568	622	678	733	785	870	918		
gap (lower/upper bound)	0%	0%	0%	0%	0%	8%	0%	27%	37%		
best rule	483	520	568	628	684	739	799	861	908	1424	2989
worst rule	483	520	568	637	716	794	881	956	1040	1940	4616
span best-worst	0%	0%	0%	1%	5%	7%	10%	11%	15%	36%	54%
FSFO SSPT[FSFO]	483	520	568	628	684	739	799	861	908	1424	2989
MACH_CONSTRAINED											
optimum	483	520	568	622							
gap (lower/upper bound)	0%	0%	0%	0%							
best rule	483	520	568	628	684	739	809	869	919	1437	3039
worst rule	483	520	568	637	716	804	905	998	1095	2085	5050
span best-worst	0%	0%	0%	1%	5%	9%	12%	15%	19%	45%	66%
FSFO SSPT[FSFO]	483	520	568	628	684	739	809	869	930	1437	3042
DUAL_CONSTRAINED											
optimum	483	524	569	640	746	838	930				
gap (lower/upper bound)	0%	1%	0%	3%	9%	19%	16%				
best rule	483	525	589	655	708	781	854	912	967	1527	3173
worst rule	483	530	620	724	823	916	999	1135	1229	2325	5500
span best-worst	0%	1%	5%	11%	16%	17%	17%	24%	27%	52%	73%
FSFO SSPT[FSFO]	483	528	598	656	708	789	875	924	979	1547	3173
OP_CONSTRAINED											
optimum	483	520	568	929	1088	1272					
gap (lower/upper bound)	0%	0%	0%	33%	38%	47%					
best rule	535	643	756	917	1013	1118	1225	1351	1456	2524	5695
worst rule	548	685	842	1041	1243	1423	1616	1807	2022	4024	10010
span best-worst	2%	6%	11%	13%	23%	27%	32%	34%	39%	59%	76%
FSFO SSPT[FSFO]	540	643	803	918	1013	1118	1229	1351	1486	2541	5695

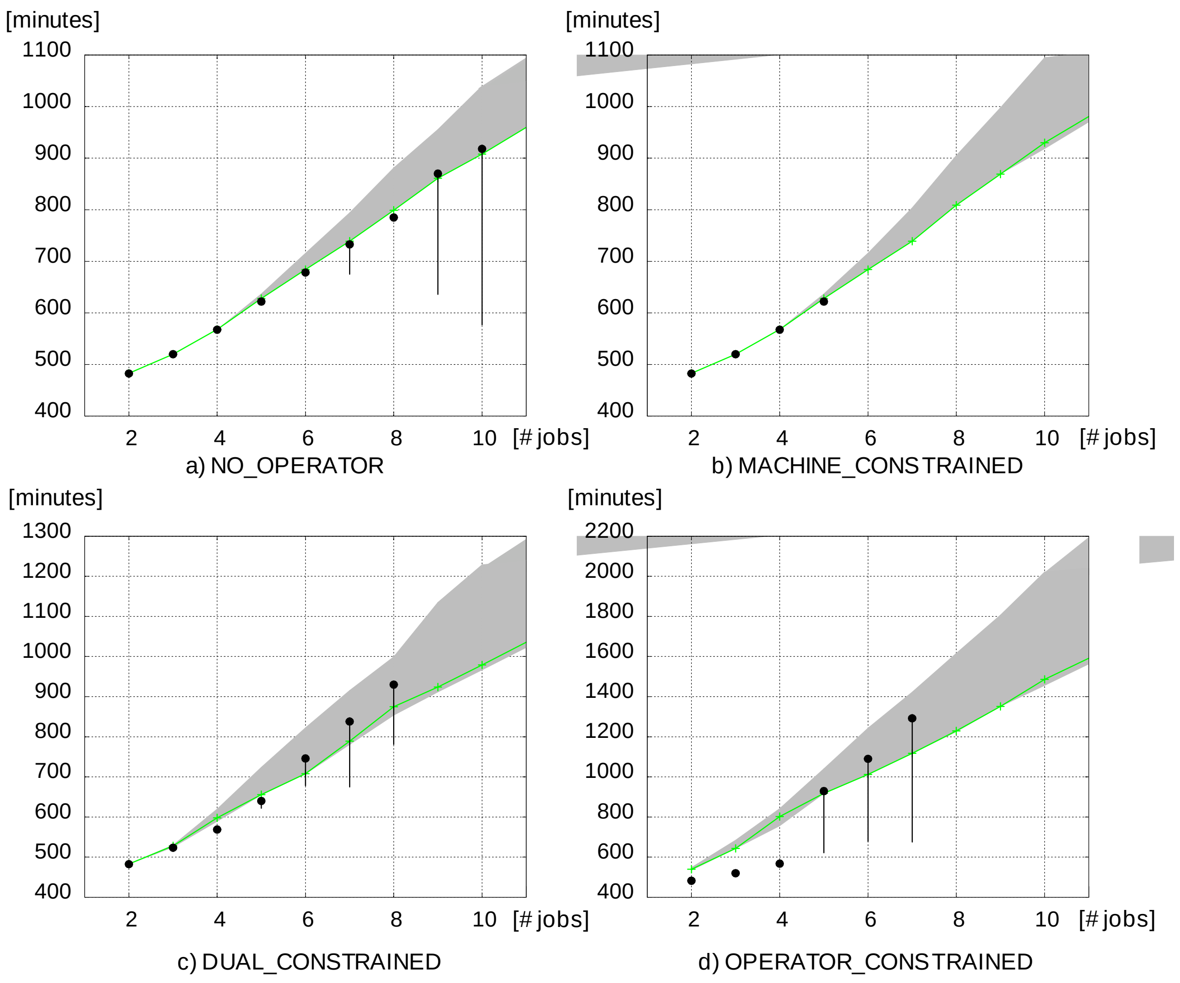

Figure 3: Results for all static cases. Green line: FSFO ShortestOpStepLength [FSFO] rule, grey area: best/worst rule combination result, black points: Solver results with gap, if any (line)

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.

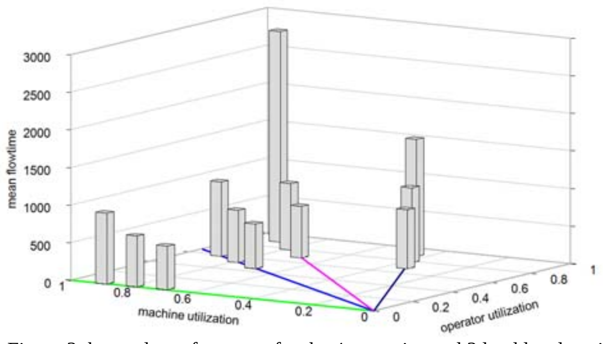

Figure 3: best rule performance for the 4 scenarios and 3 load level settings.

Table 3 lists our simulation results in more detail. Mean flow time as measured in our simulation experiment is expressed there as a ***flow factor***, i.e. flow time divided by total processing time. The best and worst rule performances for a given scenario and load level are highlighted.

To summarize our results, FSFO is the best machine rule irrespectively of the scenario and load level. Results concerning the best operator rule are a bit more complex: for the MACHINE_CONSTRAINED and OPERATOR_CONSTRAINED scenarios the operator rule SSPT[FSFO], i.e. Shortest Step Processing Time first with FSFO used as a tie breaker, yields the best results if used together with FSFO as a machine rule.

The FIFO rule performs quite badly in our experiments. In most cases it does not yield better results than random selection, i.e. choosing an arbitrary job to process next.

In the DUAL_CONSTRAINED scenario SSPT[FIFO] is the best, i.e. SSPT with tie breaker FIFO, for the 70% and 80% load levels. This changes if load is further increased to 90%. Most rule combinations are not able to handle this high load for both machines and operators anymore and the system is running full, more jobs are entering the system than leaving. In these cases no values are given in the respective table cells. Only combinations with MQL as an operator rule are able to handle this high load. The best results in this case are produced by the combination of the FSFO machine rule and MQL[FIFO] as an operator rule. The MQL – rule prefers machines with long queues, which means that operators go to high loaded machines first. That is the reason, why higher utilization level can be handled. Although this rule combination gives by far the best results for this case, the increase in average flow time is very high.

Comparing the results for MACHINE_CONSTRAINED and NO_OPERATOR, the increase in mean flow time is only small, especially for lower load levels. If only flow time estimates are of interest, it seems viable to simplify the model by ignoring the operator requirements.

The experiments with small instances show that in the OPERATOR_CONSTRAINED cases the optimal solution are much better than the results produced by heuristics. In the long term simulation the best rule results are higher for scenarios with moderate load levels of 70% and 80% compared to the DUAL_CONSTRAINED. Only the 90% load level results are smaller than its DUAL_CONSTRAINED equivalent but still about 56% higher than in the MACHINE_CONSTRAINED scenario. This seems to an effect of the heuristic procedure used (machines choose next job first, then operators choose between the machines that are ready to process). In future research more simulation runs will a different heuristic setup will be performed to analyze this effect in more detail.

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/load level combinations (but excluding DUAL_CONSTRAINED / 90%, see text); so (system overflow) more incoming than outgoing jobs.

		NO_OPERATOR			**MACH_CONSTRAINED**			**OP_CONSTRAINED**			**DUAL_CONSTRAINED**			Mean (flow factor)
		(0 Operator)			(1 Operator)			(2 Operator)			(3 Operator)			
machine rule	operator rule	utilisation of bootleneck			utilisation of bootleneck			utilisation of bootleneck			utilisation of bootleneck			
[tie breaker]	[tie breaker]	70%	80%	90%	70%	80%	90%	70%	80%	90%	70%	80%	90%	
FIFO	FIFO	*1,42*	*1,74*	*2,71*	1,49	1,94	3,71	2,26	3,40	8,26	2,22	6,28	so	3,22
FIFO	FSFO	*1,42*	*1,74*	*2,71*	1,48	1,89	3,46	1,86	2,53	4,92	1,91	3,50	so	2,49
FIFO	MQL[FIFO]	*1,42*	*1,74*	*2,71*	1,47	1,87	3,22	2,04	2,75	5,00	1,88	2,94	so	2,46
FIFO	MQL[FSFO]	*1,42*	*1,74*	*2,71*	1,47	1,86	3,21	1,88	2,56	4,76	1,85	2,91	so	2,40
FIFO	MQL[Rnd]	*1,42*	*1,74*	*2,71*	1,47	1,86	3,20	2,02	2,74	4,99	1,87	2,92	so	2,45
FIFO	Rnd	*1,42*	*1,74*	*2,71*	*1,50*	*1,95*	3,81	2,29	3,48	8,93	2,27	6,90	so	3,36
FIFO	SSPT[FIFO]	*1,42*	*1,74*	*2,71*	1,46	1,83	3,11	1,93	2,58	4,55	1,79	2,84	so	2,36
FIFO	SSPT[FSFO]	*1,42*	*1,74*	*2,71*	1,46	1,83	3,08	1,88	2,50	4,44	1,76	2,70	so	2,32
FIFO	SSPT[Rnd]	*1,42*	*1,74*	*2,71*	1,46	1,84	3,12	1,93	2,58	4,56	1,80	2,86	so	2,37
FIFO	SPT[FIFO]	*1,42*	*1,74*	*2,71*	1,47	1,85	3,22	2,04	2,84	6,02	1,82	3,03	so	2,56
FIFO	SPT[FSFO]	*1,42*	*1,74*	*2,71*	1,47	1,85	3,23	2,07	2,93	6,63	1,83	3,12	so	2,64
FIFO	SPT[Rnd]	*1,42*	*1,74*	*2,71*	1,47	1,85	3,20	2,03	2,81	5,80	1,82	3,03	so	2,54
FSFO	FIFO	**1,32**	**1,52**	**2,13**	1,35	1,59	2,30	1,99	2,69	5,15	1,71	2,63	so	2,22
FSFO	FSFO	**1,32**	**1,52**	**2,13**	**1,34**	1,58	2,26	**1,76**	2,27	3,98	1,61	2,29	so	2,00
FSFO	MQL[FIFO]	**1,32**	**1,52**	**2,13**	**1,34**	1,58	2,32	1,95	2,53	4,18	1,66	2,35	**6,56**	2,08
FSFO	MQL[FSFO]	**1,32**	**1,52**	**2,13**	**1,34**	1,58	2,33	1,82	2,37	4,01	1,62	2,28	8,40	2,03
FSFO	MQL[Rnd]	**1,32**	**1,52**	**2,13**	**1,34**	1,58	2,33	1,93	2,51	4,16	1,65	2,36	*14,65*	2,08
FSFO	Rnd	**1,32**	**1,52**	**2,13**	1,35	1,61	2,52	2,03	2,82	6,00	1,76	3,23	so	2,39
FSFO	SSPT[FIFO]	**1,32**	**1,52**	**2,13**	**1,34**	**1,56**	**2,25**	1,83	2,31	3,68	**1,54**	**2,00**	so	**1,95**
FSFO	SSPT[FSFO]	**1,32**	**1,52**	**2,13**	**1,34**	**1,56**	**2,25**	**1,76**	**2,21**	**3,51**	1,58	2,27	so	**1,95**
FSFO	SSPT[Rnd]	**1,32**	**1,52**	**2,13**	**1,34**	1,57	2,28	1,83	2,32	3,70	1,58	2,17	9,77	1,98
FSFO	SPT[FIFO]	**1,32**	**1,52**	**2,13**	1,35	1,59	2,41	1,92	2,68	7,26	1,56	2,61	so	2,39
FSFO	SPT[FSFO]	**1,32**	**1,52**	**2,13**	1,35	1,59	2,41	1,89	2,49	5,01	1,58	2,65	so	2,18
FSFO	SPT[Rnd]	**1,32**	**1,52**	**2,13**	**1,34**	1,58	2,33	1,88	2,46	4,71	1,57	3,03	so	2,17
Rnd	FIFO	*1,42*	1,73	2,69	1,49	1,93	3,80	2,26	3,41	8,63	2,23	6,85	so	3,31
Rnd	FSFO	*1,42*	1,73	2,69	1,47	1,88	3,42	1,86	2,53	4,72	1,94	3,60	so	2,48
Rnd	MQL[FIFO]	*1,42*	1,73	2,69	1,47	1,85	3,19	2,03	2,75	4,99	1,87	2,93	so	2,45
Rnd	MQL[FSFO]	*1,42*	1,73	2,69	1,46	1,85	3,17	1,89	2,57	4,79	1,85	2,91	so	2,39
Rnd	MQL[Rnd]	*1,42*	1,73	2,69	1,46	1,85	3,17	2,02	2,74	4,98	1,86	2,92	so	2,44
Rnd	Rnd	*1,42*	1,73	2,69	1,49	1,94	*3,87*	2,27	3,45	8,91	*2,29*	*7,99*	so	*3,46*
Rnd	SSPT[FIFO]	*1,42*	1,73	2,69	1,45	1,82	3,09	1,93	2,56	4,53	1,79	2,86	so	2,35
Rnd	SSPT[FSFO]	*1,42*	1,73	2,69	1,45	1,81	3,05	1,87	2,49	4,44	1,77	2,79	so	2,32
Rnd	SSPT[Rnd]	*1,42*	1,73	2,69	1,46	1,82	3,10	1,93	2,57	4,54	1,80	2,90	so	2,36
Rnd	SPT[FIFO]	*1,42*	1,73	2,69	1,46	1,84	3,19	2,03	2,84	6,14	1,84	3,24	so	2,58
Rnd	SPT[FSFO]	*1,42*	1,73	2,69	1,46	1,83	3,17	2,03	2,81	5,70	1,84	3,23	so	2,54
Rnd	SPT[Rnd]	*1,42*	1,73	2,69	1,46	1,83	3,17	2,02	2,79	5,75	1,83	3,22	so	2,54
SPT[FIFO]	FIFO	1,37	1,65	2,44	1,41	1,72	2,73	2,02	2,76	5,44	1,84	3,41	so	2,44
SPT[FIFO]	FSFO	1,37	1,65	2,44	1,41	1,73	2,72	2,08	3,24	8,09	1,93	3,67	so	2,76
SPT[FIFO]	MQL[FIFO]	1,37	1,65	2,44	1,40	1,70	2,59	2,02	2,77	5,23	1,73	2,53	13,14	2,31
SPT[FIFO]	MQL[FSFO]	1,37	1,65	2,44	1,40	1,70	2,59	1,90	2,63	5,10	1,73	2,55	13,20	2,28
SPT[FIFO]	MQL[Rnd]	1,37	1,65	2,44	1,40	1,70	2,59	2,02	2,77	5,26	1,73	2,53	14,17	2,31
SPT[FIFO]	Rnd	1,37	1,65	2,44	1,41	1,73	2,76	2,07	2,90	6,23	1,89	3,80	so	2,57
SPT[FIFO]	SSPT[FIFO]	1,37	1,65	2,44	1,40	1,70	2,59	2,12	3,17	7,04	1,74	2,69	so	2,54
SPT[FIFO]	SSPT[FSFO]	1,37	1,65	2,44	1,40	1,69	2,58	2,09	3,14	7,03	1,75	2,72	so	2,53
SPT[FIFO]	SSPT[Rnd]	1,37	1,65	2,44	1,40	1,70	2,59	2,11	3,15	6,99	1,74	2,68	so	2,53
SPT[FIFO]	SPT[FIFO]	1,37	1,65	2,44	1,41	1,72	2,69	2,35	3,77	10,48	2,12	5,00	so	3,18
SPT[FIFO]	SPT[FSFO]	1,37	1,65	2,44	1,41	1,73	2,69	*2,40*	*3,94*	*12,06*	2,15	5,23	so	3,37
SPT[FIFO]	SPT[Rnd]	1,37	1,65	2,44	1,41	1,72	2,68	2,32	3,63	9,09	2,06	4,55	so	2,99
SPT[FSFO]	FIFO	1,37	1,65	2,44	1,41	1,72	2,73	2,02	2,76	5,44	1,84	3,41	so	2,44
SPT[FSFO]	FSFO	1,37	1,65	2,44	1,41	1,73	2,72	2,08	3,24	8,09	1,93	3,67	so	2,76
SPT[FSFO]	MQL[FIFO]	1,37	1,65	2,44	1,40	1,70	2,59	2,02	2,77	5,23	1,73	2,53	13,14	2,31
SPT[FSFO]	MQL[FSFO]	1,37	1,65	2,44	1,40	1,70	2,59	1,90	2,63	5,10	1,73	2,55	13,20	2,28
SPT[FSFO]	MQL[Rnd]	1,37	1,65	2,44	1,40	1,70	2,59	2,02	2,77	5,26	1,73	2,53	14,17	2,31
SPT[FSFO]	Rnd	1,37	1,65	2,44	1,41	1,73	2,76	2,07	2,90	6,23	1,89	3,80	so	2,57
SPT[FSFO]	SSPT[FIFO]	1,37	1,65	2,44	1,40	1,70	2,59	2,12	3,17	7,04	1,74	2,69	so	2,54
SPT[FSFO]	SSPT[FSFO]	1,37	1,65	2,44	1,40	1,69	2,58	2,09	3,14	7,03	1,75	2,72	so	2,53
SPT[FSFO]	SSPT[Rnd]	1,37	1,65	2,44	1,40	1,70	2,59	2,11	3,15	6,99	1,74	2,68	so	2,53
SPT[FSFO]	SPT[FIFO]	1,37	1,65	2,44	1,41	1,72	2,69	2,35	3,77	10,48	2,12	5,00	so	3,18
SPT[FSFO]	SPT[FSFO]	1,37	1,65	2,44	1,41	1,73	2,69	*2,40*	*3,94*	*12,06*	2,15	5,23	so	3,37
SPT[FSFO]	SPT[Rnd]	1,37	1,65	2,44	1,41	1,72	2,68	2,32	3,63	9,09	2,06	4,55	so	2,99
SPT[Rnd]	FIFO	1,37	1,65	2,44	1,41	1,72	2,73	2,02	2,76	5,44	1,84	3,41	so	2,44
SPT[Rnd]	FSFO	1,37	1,65	2,44	1,41	1,73	2,71	2,07	3,18	7,67	1,88	3,33	so	2,68
SPT[Rnd]	MQL[FIFO]	1,37	1,65	2,44	1,40	1,70	2,59	2,02	2,77	5,23	1,73	2,53	13,12	2,31
SPT[Rnd]	MQL[FSFO]	1,37	1,65	2,44	1,40	1,70	2,59	1,90	2,64	5,11	1,73	2,54	13,71	2,28
SPT[Rnd]	MQL[Rnd]	1,37	1,65	2,44	1,40	1,70	2,59	2,02	2,77	5,26	1,73	2,53	14,16	2,31
SPT[Rnd]	Rnd	1,37	1,65	2,44	1,41	1,73	2,76	2,07	2,90	6,23	1,89	3,80	so	2,57
SPT[Rnd]	SSPT[FIFO]	1,37	1,65	2,44	1,40	1,70	2,59	2,12	3,17	7,04	1,74	2,69	so	2,54
SPT[Rnd]	SSPT[FSFO]	1,37	1,65	2,44	1,40	1,69	2,58	2,09	3,14	7,03	1,75	2,73	so	2,53
SPT[Rnd]	SSPT[Rnd]	1,37	1,65	2,44	1,40	1,70	2,59	2,11	3,15	6,99	1,74	2,68	so	2,53
SPT[Rnd]	SPT[FIFO]	1,37	1,65	2,44	1,41	1,72	2,69	2,35	3,77	10,48	2,12	5,00	so	3,18
SPT[Rnd]	SPT[FSFO]	1,37	1,65	2,44	1,41	1,72	2,69	2,33	3,64	8,99	2,10	4,78	so	3,01
SPT[Rnd]	SPT[Rnd]	1,37	1,65	2,44	1,41	1,72	2,68	2,32	3,63	9,09	2,06	4,55	so	2,99
flowfactor best rule		**1,32**	**1,52**	**2,13**	**1,34**	**1,56**	**2,25**	**1,76**	**2,21**	**3,51**	**1,54**	**2,00**	**6,56**	**1,95**
flowfactor worst rule		*1,42*	*1,74*	*2,71*	*1,50*	*1,95*	*3,87*	*2,40*	*3,94*	*12,06*	*2,29*	*7,99*	*14,65*	*3,46*
flowtime span ((worst-best)/best)		8,2%	14,7%	27,3%	12,2%	24,8%	72,5%	36,0%	78,0%	243,0%	48,6%	299,6%	123,5%	77,4%

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

[1] R. Haupt, "A survey of priority rule-based scheduling," *OR Spectrum*, vol. 11, pp. 3–16, March 1989.
[2] G. Fisher, H; Thompson, "Probabilistic learning combinations of local job-shop scheduling rules," *Industrial Scheduling; J.F. Muth, G.L. Thompson(Hrsg.); Englewood Cliffs*, pp. 225 – 251, 1963.
[3] J. Adams, E. Balas, and D. Zawack, "The shifting bottleneck procedure for job shop scheduling," *Management Science*, vol. 34, no. 3, pp. 391–401, 1988.
[4] P. J. M. v. Laarhoven, E. H. L. Aarts, and J. K. Lenstra, "Job shop scheduling by simulated annealing," *Operations Research*, vol. 40, no. 1, pp. 113–125, 1992.
[5] C. Zhang, P. Li, Z. Guan, and Y. Rao, "A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem," *Computers & Operations Research*, vol. 34, no. 11, pp. 3229 – 3242, 2007.
[6] H. Zhou, W. Cheung, and L. C. Leung, "Minimizing weighted tardiness of job-shop scheduling using a hybrid genetic algorithm," *European Journal of Operational Research*, vol. 194, no. 3, pp. 637 – 649, 2009.
[7] A. Manikas and Y. Chang, "Multi-criteria sequence-dependent job shop scheduling using genetic algorithms," *Computers & Industrial Engineering*, May 2008.
[8] B. Scholz-Reiter, M. Görges, and T. Philipp, "Autonomously controlled production systems–influence of autonomous control level on logistic performance," *CIRP Annals - Manufacturing Technology*, vol. 58, no. 1, pp. 395 – 398, 2009.
[9] J. H. Blackstone, D. T. Phillips, and G. L. Hogg, "A state-of-the-art survey of dispatching rules for manufacturing job shop operations," *International Journal of Production Research*, vol. 20, no. 1, pp. 27–45, 1982.
[10] C. Rajendran and O. Holthaus, "A comparative study of dispatching rules in dynamic flowshops and jobshops," *European Journal of Operational Research*, vol. 116, pp. 156–170, July 1999. dispatching rules, comparison, job shop, flow shop.
[11] C. Geiger, R. Uzsoy, and H. Aytu?, "Rapid modeling and discovery of priority dispatching rules: An autonomous learning approach," *Journal of Scheduling*, vol. 9, pp. 7–34, Feb. 2006. genetic programming, single machine.
[12] C. D. Geiger and R. Uzsoy, "Learning effective dispatching rules for batch processor scheduling," in *International Journal of Production Research* geiger2006, pp. 1431–1454. batching, genetic programming, single machine.
[13] K.-C. JEONG and Y.-D. KIM, "A real-time scheduling mechanism for a exible manufacturing system: using simulation and dispatching rules," 1998.
[14] V. B. Gargeya and R. H. Deane, "Scheduling research in multiple resource constrained job shops: a review and critique," *International Journal of Production Research*, no. 34, pp. 2077–2097, 1996.
[15] Y. MATI and X. XIE, "A genetic-search-guided greedy algorithm for multi-resource shop scheduling with resource flexibility," 2007.
[16] S. Dauzère-Pérès, W. Roux, and J. B. Lasserre, "Multi-resource shop scheduling with resource flexibility," *European Journal of Operational Research*, vol. 107, no. 2, pp. 289 – 305, 1998.
[17] S. Dauzère-Pérès and C. Pavageau, "Extensions of an integrated approach for multi-resource shop scheduling," 2003.
[18] E. H. Patel, V. and I. Ben-Abdallah, "Scheduling in dual-resources constrained manufacturing systems using genetic algorithms," 1999.
[19] P. B. Chen, Dong ; Luh, "Optimization-based manufacturing scheduling with multiple resources, setup requirements, and transfer lots," *IIE Transactions*, vol. 35, pp. 973 – 985, 2003.
[20] H. ElMaraghy, V. Patel, and I. B. Abdallah, "Scheduling of manufacturing systems under dual-resource constraints using genetic algorithms," *Journal of Manufacturing Systems*, vol. 19, no. 3, pp. 186 – 201, 2000.
[21] R. L. Gerry Feigin, John Fowler, "Masm test data sets. (2009). http://www.eas.asu.edu/ masmlab. last accessed on 28th may 2009."
[22] K. S. . T. Mohamed K. El Adl Annando A. Rodriguez, "Hierarchical modeling and control of re-entrant semiconductor manufacturing facilities," 1996.
[23] ILOG, "Cplex - mathematical programming optimizers." http://www.ilog.com/products/cplex/.
[24] J. C.-H. Pan and J.-S. Chen, "Mixed binary integer programming formulations for the reentrant job shop scheduling problem," *Comput. Oper. Res.*, vol. 32, no. 5, pp. 1197–1212, 2005.
[25] A. M. Law and D. W. Kelton, *Simulation Modelling and Analysis*. McGraw-Hill Education - Europe, April 2000.