An adaptive scheduling framework for solving multi-objective hybrid flow shop scheduling problems

Abdulrahman Nahhas
Faculty of computer science
Otto von Guericke University
Magdeburg, Germany
abdulrahman.nahhas@ovgu.de

Marco Krist
Tectron Worbis GmbH
Osterode am Harz, Germany
marco.krist@tectron-worbis.de

Klaus Turowski
Faculty of computer science
Otto von Guericke University
Magdeburg, Germany
klaus.turowski@ovgu.de

Abstract

The proposed new technologies in the context of industry 4.0 challenge the current practices of scheduling in industry and their associated research in academia. The conventional optimization techniques that are employed for solving scheduling problems are either computationally expensive or lack the required quality. Therefore, in this paper, we propose an adaptive scheduling framework to address scheduling problems taking into account multi-objective optimality measures. The framework is motivated by a hybrid design to combine the use of heuristic and metaheuristic approaches. The main idea behind the presented concept is to achieve an acceptable tradeoff between the quality of the suggested solutions for a problem and the required computational effort to obtain them. The perused narrative in such implementation is combining some advantages of heuristic and metaheuristic approaches such as: the light execution time of heuristics and the robustness as well as the quality of metaheuristic approaches. The framework is evaluated for solving hybrid flow shop scheduling problems that are derived from a real use case.

1. Introduction

The emerging concepts of industry 4.0 enabled many appealing opportunities as well as new challenges. These have significant impacts on the strategic and operative management activities of manufacturing enterprises [5]. However, to achieve the current visions of industry 4.0, practitioners and academics need to commit to fundamental modifications to the traditional practices in the industry and the associated research in academia. Although profound steps toward the digitization of the industrial environment have been accomplished, the current implementation of industry 4.0 projects still exhibits strong practical nature with insufficient research efforts [29]. Among core management processes in any enterprise is scheduling activities since they intersect with many strategic and operative levels of operation in any manufacturing environment. One of the main challenges that are evident by the adoption of the industry 4.0 technologies is the instant required reaction to changes in the recorded system state [32]. Energetic reaction to different events in a manufacturing system allows to exploit several optimization potentials on both operative and strategic levels through optimized scheduling. The optimization potential increases when the system is characterized by high variety in product types and shorter lead times. In addition, the disruptions in supply chain (e.g. a late delivery of raw material that are used for producing some product type) would possibly require major modification of production planning, which could be carried out as quick as possible.

One of the main data streams in the context of scheduling policies is machine breakdowns. For instance, Nahhas et al. [29] investigated the optimization potential of different industry 4.0 concepts and concentrated on scheduling problems. The authors studied hybrid flow shop scheduling problems taking into consideration the impact of including machine breakdowns in the optimization. They concluded in their findings that considering machine breakdowns during the optimization is not recommended since the computational effort to solve the problems significantly increased. Therefore, a thorough investigation of adaptive solution techniques is suggested to propose new frameworks for dealing with scheduling problems with light execution time while maintaining high solution quality. In this research, we present an adaptive scheduling framework that is inspired by a hybrid design to address scheduling problems with light execution time. The framework is designed to solve scheduling problems and deliver high solution quality with relatively less required computational effort in comparison to the conventional stat of the art metaheuristic optimization techniques. The evaluation of the framework is based on real-world problems that are extracted from the production log of a manufacture in the field of print circuit board assembly production.

URI: https://hdl.handle.net/10125/70811
978-0-9981331-4-0

The presented approach is compared to Genetic Algorithms for solving the problems. In addition, the solution approaches presented in Aurich et al. [3] and later compared against Genetic Algorithms (GA) in Nahhas et al. [28] for solving a two-stage Hybrid Flow Shop (HFS) are reconstructed and compared against the presented framework. An HFS scheduling problem involves several processing stages. On every stage at least two parallel machines are available to process all jobs. In addition, all jobs must follow the same technological order to be processed on different stages [31]. In the course of the presented paper, a mathematical formulation of the considered problem is presented in the second section. The problem is formulated based on a thorough analysis of the investigated production environment. The third section comprises an overview of the state-of-the-art approaches that are often used to solve HFS scheduling problems. Followed in the fourth section, a novel adaptive scheduling framework is proposed. The evaluation of the presented framework for solving thirty problem instances is presented in the fifth section. Finally, the paper is closed with some suggestions and further research directions.

2. Problem statement

2.1. System description

The presented scheduling framework is evaluated based on the case study, which is derived from the field of Printed Circuit Board (PCB) assembly production. A PCB usually go through four main processing procedures. The first operation is carried out on the so-called Surface Mounting Device (SMD) machines. In this stage, hundreds of components are mounted on the surface of an empty PCB. This processing stage is characterized by major and minor family sequence-dependent setup times. The first investigations of family sequence-dependent setup time can be found in [20, 38]. The surface mounting process of PCB is highly automated. Therefore, the setup process of machines significantly impacts the level of system efficiency [38]. Accordingly, considering the number of major setup times is crucial to enhance the overall system utilization.

The nature of the production in such manufacturing environments is highly customized, in which hundreds of part types might be demanded. In turns, setting scheduling policies in such environment is even more complicated. Therefore, the part types that share raw materials, other properties, and operational procedures, are clustered into groups or families [38]. Jobs with different part types require different processing time on the different processing stages, while only minor setup times (20 minutes) are required to configure the machines. However, jobs with different family types provoke major setup time (45-120 minutes) to configure the machines when switching from one to another. The investigated production environment, five parallel machines with different speeds are available to process jobs.

The second operation is performed on Automated-Optical-Inspection (AOI) quality control machines. In this processing stage, the PCBs undergo different quality tests to ensure that the components are placed correctly. In the investigated system, five parallel machines with different speeds are used to process jobs on the AOI processing stage. The third operation is performed using Selective Soldering (SS) machines. In the investigated system many jobs must undergo the third stage, where five parallel machines are available. Jobs are scheduled with sequence-independent minor setup times on the machines in the second and third stages. The fourth and final operation is the conformal coating process, which is performed using two identical parallel Conformal Coating (CC) machines with family major and minor setup times. The structure of the investigated system is presented in Figure 1.

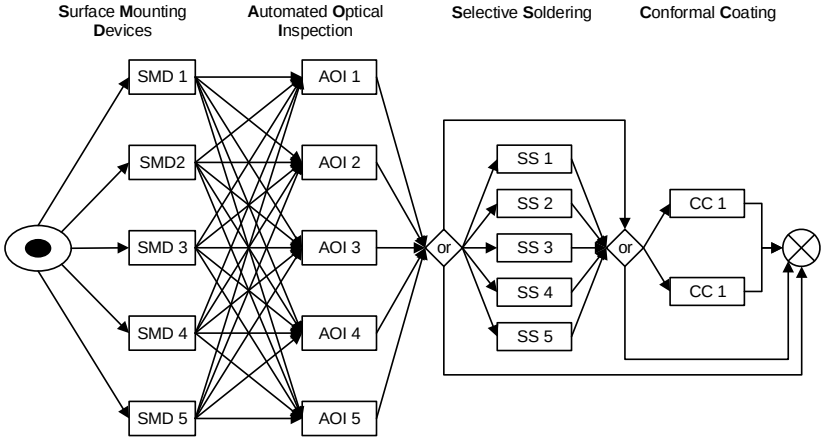

Figure 1. The investigated production environment.

2.2. Problem formulation

The required notions and preliminaries for the mathematical formulation of the problem is presented in the following:

- Let $J \in \{J_j, \ldots, J_n\}$ denotes a set of n jobs ($j = 1, \ldots, n$) that are released for scheduling.
- Let $S = \{S_1, \ldots, S_s\}$ denotes a set of s processing stages $(s \in \{1, \ldots, 4\})$ that contain a set M_s of parallel machines on each.
- Let $M_{i,s} \in M_s = \{M_1, \ldots, M_{m_s}\}$ denotes a machine in the set M_s of m machines on processing stage s $(i \in \{1, \ldots, m_s\})$.
- Let d_j $(d_j \in \{1, \ldots, 20\})$ denotes the days left to the due date of a job J_j which corresponds to the priority of the job.
- Let C_j denotes the completion time of a job in J_j
- Let T_j be the recorded tardiness of a job in J_j where: $T_j = \begin{cases} (C_j - d_j)\ if\ (C_j - d_j) > 0 \\ 0\ Otherweise \end{cases}$
- Let U_j be the associated unit penalty of a tardy job in J_j where: $U_j = \begin{cases} 1\ if\ C_j > d_j \\ 0\ Otherweise \end{cases}$
- Let $\gamma_1 = C_{max},\ max\, C_j : \forall J_j\ (j \in \{1, \ldots, n\})$ denotes the makespan or the maximum completion time of the set of job *J*.
- Let $\gamma_2 = MS \in \{0, \ldots, n-1\}$ denotes the number of required major setup times on the first stage to process all jobs.
- Let $\gamma_3 = T,\ T = \sum_{j=1}^{n}, T_j : \forall J_j\ (j \in \{1, \ldots, n\})$ be the total tardiness of all jobs.
- Let $\gamma_4 = U,\ U = \sum_{j=1}^{n} U_j : \forall J_j\ (j \in \{1, \ldots, n\})$ be the total number of recorded penalties of all jobs.

Let $\mathbb{H}$ be the set of all production schedules for a set of jobs *J*. The goal is to obtain a production schedule $H \in \mathbb{H}$. This production schedule is subject to the minimization of the makespan, the necessary number of major setup times to complete all jobs, the total tardiness, and the number of penalties as demonstrated in formula (1).

$$\begin{gathered} \gamma_1(H) = C_{max}\,,\ \gamma_2(H) = MS \\ \gamma_3(H) = \sum_{j=1}^{n} T_j\,,\ \ \gamma_4(H) = \sum_{j=1}^{n} U_j \\ min\, Z(H) \Leftrightarrow min\, \gamma_1(H) \wedge min\, \gamma_2(H) \wedge \\ min\, \gamma_3(H) \wedge min\, \gamma_4(H) : \forall\, H \in \mathbb{H} \end{gathered} \tag{1}$$

To further formulate the objective function, a weighted sum approach has been adapted as shown in formula (2).

$$\arg\min_{H \in H} Z(H) = W_1 \cdot \gamma_1 + W_2 \cdot \gamma_2 + W_3 \cdot \gamma_3 + W_4 \cdot \gamma_4$$
$$:\forall\ (W_1 + W_2 + W_3 + W_4 = 1),\ W \geq 0 \tag{2}$$

The assumptions and operational constraints that are subject to this problem formulation are listed in the following:

- The number of jobs during the considered scheduling period is known and fix.
- The processing times of jobs on different stages are known and fix.
- A job can be processed on only one machine at the same time.
- Preemption of jobs is not allowed.
- Jobs that belong to the same family cannot be processed on different machines at the same time.
- A machine can process only one job at the same time.
- The buffer capacity between processing stages is assumed to be unlimited.

3. Related works

Although setting scheduling policies is an operative task it has a profound impact on major strategic decision-making processes, which are directly linked to operational costs [34]. From an academic point of view, their challenging and complex nature has been an interesting puzzle for many scholars. However, the majority of those puzzles have been proven to be NP-Hard combinatorial optimization problems [11, 22]. This implies that with the currently available computational power and advances, it is implausible to develop polynomial algorithms that can deliver optimal solutions.

Nevertheless, some implementations of the exact optimization methods as for instance branch and bound [7] or dynamic programming [14] have been proposed for solving small size scheduling problems. Although such solution techniques guarantee optimal or bounded optimal solutions, their required computational effort can easily grow exponentially for solving complex scheduling problems. For a comprehensive discussion about the adoption of exact methods for solving Hybrid Flow Shop (HFS) scheduling problems, one can refer to the contribution of Kis and Pesch [18].

In reality, scheduling activities are usually carried out based on experiences, intuitions, and well-established constructive policies. These practices formed another research stream in the scientific community that deals with the so-called Priority Dispatching Rules (PDRs). PDRs are basically simple

constructive heuristics that are mostly based on computing some index to priorities jobs and dispatching them for production. The Earliest Due Date (EDD), the Longest Processing Time (LPT), and the Shortest Processing Time (SPT) are some examples of such simple constructive heuristics. The EDD as the name implies is used to minimize the total tardiness and incurred penalties of jobs. The SPT is widely used for the minimization of the makespan C_{max} and/or the mean flow time as suggested in [15]. However, the oversimplified design of PDRs usually overlooks many crucial aspects of an investigated problem, which lead to potential loss of optimization opportunities. Based on this simple discussion, one can notice the evident gap between the research conducted on HFS scheduling problems and scheduling in practice [34].

The anticipated middle ground between these two directions in scheduling is improvement heuristic and metaheuristic optimization techniques. Improvement heuristics are complex heuristics that inherit iterative optimization behavior. After constructing an initial solution for a problem, an improvement procedure is designed to conduct systematic modifications on it to seek some improvement for optimizing some objective function [37]. Some of the earliest contributions in the HFS scheduling fields are presented by Wittrock [38], Gupta [12], and Voss [36]. Wittrock addressed the identical parallel machines scheduling problem for minimizing the makespan with major and minor setup times. A similar investigation is presented in [20]. A fairly more complicated two stages HFS with a single machine on the second stage was investigated by Gupta [12] and later improved by Voss [36]. The authors also perused minimizing the makespan. Some similar investigations on the two-stages HFS scheduling problems can be found in [13, 23, 30]. However, the majority of these contributions address HFS scheduling problems to minimize a single objective measure as pointed out also by Ruiz [34]. For solving multi-objectives HFS scheduling problems, metaheuristics are the dominant adopted solutions techniques. These techniques proved their superiority over conventional heuristics for solving very complex combinatorial optimization problems [9].

The majority of metaheuristic approaches are based on mimicking some natural phenomena as for instance: Simulated Annealing (SA) [17] (annealing process of metal), Genetic Algorithms (GA) [10] (evolution theory), Swarm Intelligence [24] (swam behavior), and several other evolutionary algorithms [16]. Generally, every metaheuristic approach consists of two main components: A heuristic search algorithm and an overall control strategy that guide this heuristic. The prevalence adoption of metaheuristic techniques can be traced back to their ability to seek solutions that are subject to multi-objective optimality measures. In addition, the population-based metaheuristics are able to conduct a broad and extensive search in the solution space of a problem. Thus, they report significant performance for solving complex HFS as presented in [2, 3, 6, 19, 27, 30, 35].

Some multi-objective optimization of the HFS scheduling problems was presented in Aurich et al. [3] and later compared against Genetic Algorithms (GA) in Nahhas et al. [28]. In those papers, the authors investigated a two-stage Hybrid Flow Shop (HFS) scheduling problem with family sequence-dependent setup times. Their problem formulation targets the minimization of multi-objective optimality measures. They presented a comparison between well-known Priority Dispatching Rules (PDRs), a heuristic named ISBO, and conventional metaheuristic optimization techniques such as SA [17], Tabu Search [8] and later GA in [28]. They reported outperformance in terms of minimizing the makespan and the total number of major setup times using the ISBO. However, a clear dominance could not be concluded, since the presented approach failed to outperform the metaheuristic approaches in terms of minimizing the total tardiness. As mentioned earlier, we will reconstruct the presented heuristic in [3] and compare its performance against the proposed framework. The most recent investigation of the problem is presented in [21]. The authors successfully applied neural networks for solving the problem and compared their results against the solutions presented in [3]. The reported results showed an outperformance of the presented concept for solving the problem to minimize the makespan with a slight deviation for minimizing the total tardiness. However, the reported number of required major setup is rather high.

The investigated problem was proven to be NP-hard and is less complicated to the considered problem in this paper. In this paper, we deal with a four-stage HFS scheduling problem with parallel machines that have different speeds on the processing stages. Based on the conducted analysis of the related works in the literature, the majority of the presented conventional solutions for solving HFS scheduling problems exhibit either concrete or generic nature. The specifically designed heuristics are effective to solve moderate size problems, in which a single objective is usually perused such as the majority of Priority Dispatching Rules (PDRs) [15]. Thus, such algorithms often fail to address the various needs and objectives of real industries. In addition, advanced constructive heuristics such as in [3, 38] usually require a thorough analysis of the considered problem to be accordingly designed. Accordingly, their complicated structure is very tedious to modify for addressing even minor changes in a considered system.

4. A novel adaptive scheduling framework

4.1. A conceptual model and the main components of the framework

To address the drawbacks of conventional heuristic and metaheuristic approaches, we investigate hybrid optimization strategies to present a near real-time scheduling framework that delivers at the same time solutions with high-quality. In Figure 2, an adaptive scheduling framework that utilizes the light execution time of heuristic approaches and the robustness of metaheuristic approaches is proposed. The framework consists of two main components that are linked to a conventional cyber representation of the production system. The core component of the presented framework is the adaptive scheduling component. This component basically collects problem-data and have access to:

- Possible performance and objective measures (e.g. Makespan, due dates and total tardiness).
- Operational constraints (e.g. family operational constraint, buffer capacities between processing stages, raw materials constraints).
- Real-time events that could trigger a rescheduling process (e.g. arrivals of new high priority jobs, long machine breakdowns).
- Different scheduling algorithms
- Feedback loops from the data analytics component to adjust the performance of the optimization model.

Operational constraints might be extracted through analyzing historical and real-time data streams. Machine breakdowns are an example of such data streams that could be predicted to achieve planned maintenance strategy using fuzzy logic as suggested in [1]. Thus, such breaks in the production schedule could be included in the optimization in form of fuzzy rules to achieve higher stability with more accurate solutions. The adaptive scheduling component consists of two subcomponents, namely, the optimization model and the evaluation model.

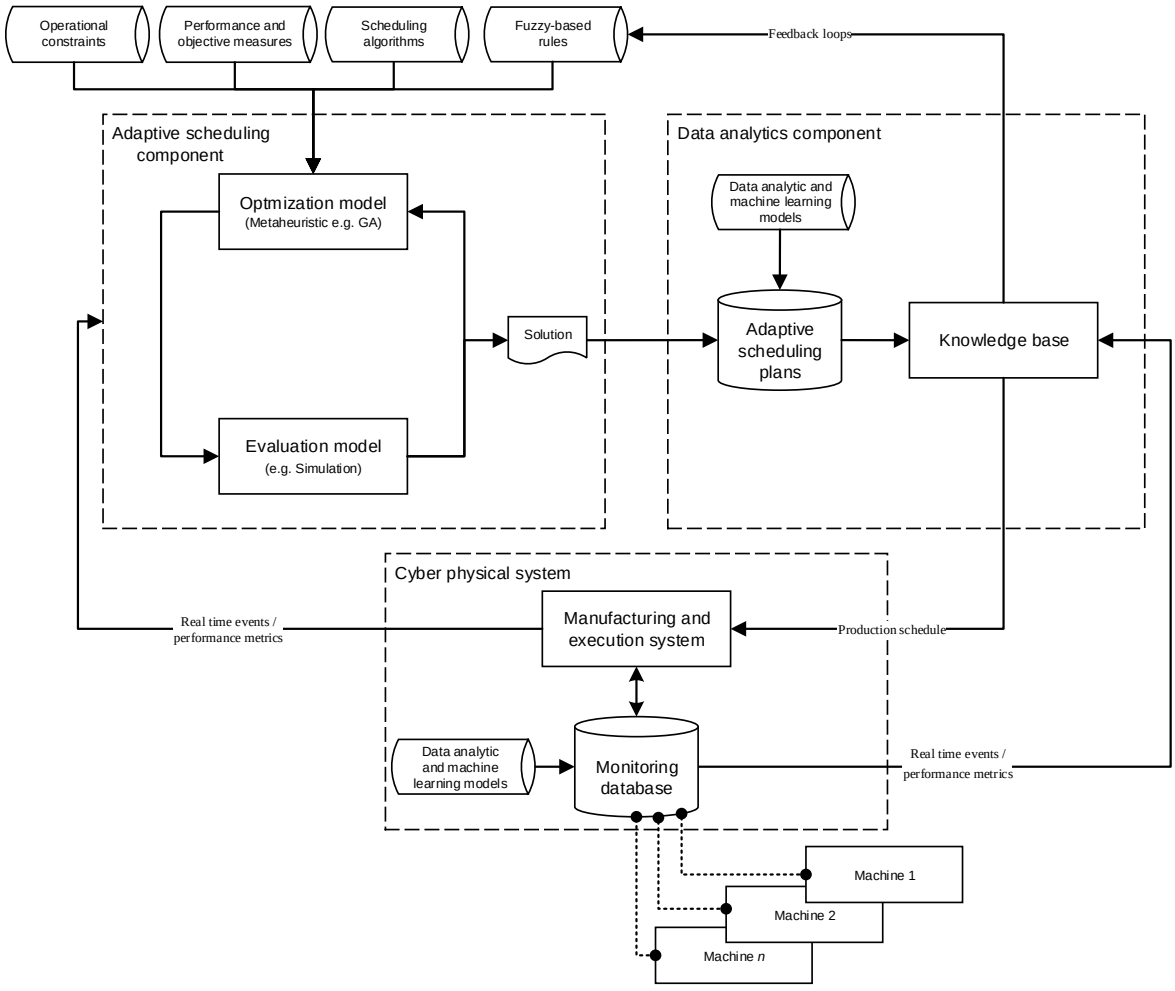

Figure 2. An adaptive scheduling framework.

In the presented framework, we relied on Genetic Algorithms (GA) [10] to design the optimization model. Furthermore, the evaluation model is based on discrete event simulation, which is used to evaluate the fitness of the solutions that are proposed by the optimization model. The adoption of GA is motivated by their broad and common adoption for solving optimization problems in scheduling literature [19, 26, 28, 33, 34]. The optimization model can have access to many state of the art heuristics, Priority Dispatching Rules (PDRs) and/or other specifically designed algorithms that are used for solving scheduling problems. In our prototypical implementation, we included four self-developed simple heuristics, which we will briefly discuss in the coming subsection. The framework is designed to present solutions that are subject to multi-objective optimization as mentioned earlier in the mathematical formulation of the problem. Thus, to address the requirements and various objectives of real-world manufacturing environments in contrast to the majority of the contributions in the HFS literature [34]. In the presented evaluation of the framework, we formulated a multi-objectives optimization problem considering two main accepts:

- The system efficiency and utilization level: through the minimization of the makespan and the total number of major setup times, which is required to process all jobs.
- The customer satisfaction: through the minimization of the total tardiness and the associated recorded penalties in delivery appointments of jobs.

The internal design of the optimization model is based on using many heuristics that are controlled by a metaheuristic approach during the scheduling interval for solving the problem. We argue that scheduling policies need to be adjusted with respect to changes in the system state to deliver better production schedules. This implies that simulating the use of different scheduling policies overtime considering simulated system states would allow achieving a higher optimization potential. In turn, we solve the scheduling the problem using an indirect encoding. The GA is encoded to switch between different heuristics overtime during the simulated scheduling period. The encoding of the GA algorithms will be discussed in detail in the coming subsections.

The role of the data analytics component is to analyze different data streams to provide feedback loops to adjust the optimization. Data streams such as machine breakdowns, arrivals of new jobs, inventory levels of required raw materials, major deviations between suggested production plans and actual ones can be analyzed to derive rules for adjusting the optimization model. Production environments are inherently associated with high structural complexity. The goal of this component is to analyze certain deviations of the proposed production plans from the actual executed plans. These uncertainties can be addressed by the adaptive component in the next optimization run through adjusting certain operational constraints (e.g. adjusted planned maintenance schedules) and other sensitive parameters. In addition, one can probably investigate training some machine learning models on the obtained solutions to extract knowledge. Here it is of a major interest to investigate whether machine-learning models can be used to solve scheduling problems based on collected historical solutions of some optimization techniques? The evaluation of this component is however not in the scope of this paper due to the extensive required analysis on the obtained solutions.

4.2. The design of the solution strategy

Dealing with Hybrid Flow Shop (HFS) scheduling problems includes solving two main sub problems namely, the allocation and the sequencing problems. Solving an HFS scheduling problem fundamentally involves answering two main questions:

- How jobs should be allocated to the available machines in every processing stage?
- What is the best sequence to process allocated jobs on every machine to satisfy some objective values?

Usually dealing with these sub problems independently is a very common practice as suggested in [4, 31] and conducted in [3, 12, 28, 36]. The goal is, eventually, to reduce the complexity of an investigated problem. In the presented evaluation of the concept, the allocation and sequencing parts of the problem are also solved independently. This implies that after allocating jobs to SMD machines, five single machine scheduling problems are to be solved. For solving the allocation part of the problem, we argue that using several algorithms would outperform using a single specifically designed heuristic or a robust metaheuristic over a determined scheduling period. To know exactly which algorithm must be used at which point in time during the scheduling period, Genetic Algorithms (GA) are used.

For solving the sequencing part of the problem on the machines, the sequencing algorithm presented in [3, 28] is adopted. Adopting this algorithm is crucial to maintain a fair comparison between the presented framework and the algorithms suggested in [3, 28]. Briefly, the sequencing algorithm is designed to dispatch jobs taking into account the tradeoff between the number of major setup times and the priority of jobs. For dispatching jobs on the second, third and fourth processing stages, the Earliest Due Date rule is used to minimize the total tardiness.

4.3. Genetic algorithms and problem encoding

The encoding of the genetic algorithms targets the problem indirectly by optimizing the selection of different algorithms to solve the scheduling problem. In this context, a solution candidate in the population of genetic algorithms is a vector of integer values that contains indexes of the included heuristics at predefined points in time T_n. We set the optimization to select two heuristics per day for twenty days scheduling period for solving the allocation part of the problem. In addition, we also encoded the GA for directly solving the allocation problem through allocating the families to the available machines in the first processing stage. A representation of a solution individual of the hybrid framework and pure GA is presented in Figure 3. At the beginning of the optimization, a random set of solution individuals is generated to form the first population of the GA. Thereafter, every solution individual is evaluated using the simulation model based on the objective function, which is presented in the mathematical formulation (see. Section 2.2).

Based on the assigned fitness values, a tournament selection strategy is adopted to pick the parents for evolving a new generation of solutions. The decision to adopt tournament selection is motivated by the ability to select solutions with low quality to generate the new offspring. In turn, this practice ensures maintaining higher diversity in the generated solution candidates and can contribute to avoiding being trapped in local optima [25]. The tournament selection strategy has been profoundly discussed in [25]. After the selection process, a uniform crossover and random mutation operators are used to mix the genes of the parents and pass them to the next generation. Finally, elitism strategy is implemented in this GA to ensure that the best solution candidates survive to the next generation [19]. After the evolving process, the new population is passed to the simulation model to investigate their fitness. This process is repeated until the optimization converges. We formulated a simple convergence function based on the relative distance between the best and the worse solution candidates using the mean of their fitness in the current population.

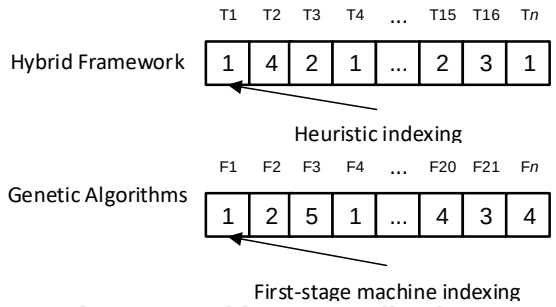

Figure 3. Problems encoding in GA.

4.4. Included heuristics

In the presented evaluation of the framework, we included four simple heuristics to be used for solving the allocation part of the problem in the first processing stage. All allocation heuristics are targeting the allocation problem on a family level. This practice is enforced as an operational constraint by the manufacture, since preparing two machines for processing jobs from the same family is not possible in the meantime. We initially tested eight heuristics and the optimization converged avoiding four of them. A simple description of the included heuristic is briefly discussed in the following:

1. *Total family first fit ascending:* After accumulating the processing time of jobs per family, the families are sorted in ascending order. Additionally, the SMD machines are also sorted in ascending order with respect to their current workload. Then, the first family is allocated to the first SMD machine (lowest loaded machine) before finally updating the load of the machine. This process is iteratively conducted until all families are allocated.
2. *Partial family first fit ascending:* Similarly, the processing time of jobs with the next highest five priorities are accumulated per family. Then, the families are sorted in ascending order. Additionally, the SMD machines are also sorted in ascending order with respect to their current workload. Then, the first family is allocated to the first SMD machine (lowest loaded machine) before finally updating the load of the machine. This process is iteratively conducted until all families are allocated.
3. *Highest priority first fit ascending*: After accumulating the processing time of jobs per family. The families are then sorted in ascending order with respect to the priorities of their jobs. Similarly, machines are sorted in ascending order based on their current workload. Then the first family is allocated to the first machine before updating the load of the machine. This process is repeated until all families are allocated.
4. *Highest priority-Smallest family first fit ascending:* After accumulating the processing time of jobs per family. The families are then sorted in ascending order based on the accumulated processing time. Once again, the families undergo a second ascending sorting based on the priorities of their jobs. Similarly, machines are sorted in ascending order based on their current workload. Then the smallest family that contains the highest priority is allocated to the first machine before updating the load of the machine. This process is repeated until all families are allocated.

5. Evaluation and computational results

Thirty HFS scheduling problem instances are solved using the proposed framework and compared with pure GA, the proposed algorithm in [3, 28], and the EDD rule. The GA is encoded for solving the problem through allocating the families to the available machines in the first processing stage. We set the population size of the GA to 15 and used a 0.4 mutation rate. The parameters of the GA are obtained empirically based on initial analysis. The desired solutions are subject to the minimization of the makespan C_{max} (minutes) the total number of major setup time required to process all jobs *MS*, the total tardiness over all jobs *T* (minutes) and the associated number of recorded penalties *U*. The used weights are 0.2, 0.2, 0.4 and 0.2 respectively. For calculating the fitness value of the proposed solution candidate by the GA, we normalized the objective values to a range between zero and one. The normalization is necessary since the considered objective values have different natures. The results for solving thirty problem instances are shown in Table 1. The EDD delivers good results to minimize the total tardiness and the recorded number of penalties. However, it fails to report even acceptable results for minimizing neither the makespan nor the total number of major setup times.

Table 1. The computational results for solving thirty problem instances.

	Hybrid Framework				GA				ISBO				EDD			
P	C_{max}	MS	U	T	C_{max}	MS	U	T	C_{max}	MS	U	T	C_{max}	MS	U	T
1	17595	51	0	0	19056	52	0	0	17807	49	4	1021	19685	132	1	121
2	14247	40	0	0	15691	42	0	0	14380	41	5	2134	16339	134	1	790
3	16680	48	0	0	16886	50	0	0	16735	48	3	1413	18766	127	0	0
4	18234	48	0	0	19281	56	0	0	18432	47	4	652	19890	130	0	0
5	16745	42	0	0	17928	44	0	0	16785	44	6	2351	18694	121	1	49
6	17413	51	0	0	17648	50	0	0	18904	47	5	1546	20182	127	3	605
7	16483	42	0	0	16834	43	0	0	16620	43	3	473	17602	129	0	0
8	16997	46	0	0	20467	46	0	0	17575	43	4	1936	18124	135	0	0
9	16349	46	0	0	15658	46	0	0	16425	48	2	354	16703	134	0	0
10	14288	40	0	0	15936	40	0	0	14708	40	2	206	16876	132	0	0
11	18159	51	0	0	20522	57	2	500	18279	47	4	1900	19546	134	1	1499
12	15632	43	0	0	19232	43	0	0	16116	40	4	1994	17596	129	0	0
13	12358	39	0	0	14606	39	0	0	12941	38	2	886	15438	133	0	0
14	16099	43	0	0	17605	42	0	0	16026	43	2	816	17899	128	0	0
15	15098	38	0	0	16691	41	0	0	15480	40	0	0	16224	134	0	0
16	15420	40	0	0	15918	46	0	0	15478	43	4	586	16699	139	0	0
17	16330	40	0	0	18886	43	0	0	16359	41	4	1888	18817	135	0	0
18	16514	47	0	0	19767	46	0	0	16584	47	2	835	17994	128	0	0
19	17059	47	0	0	17921	50	0	0	17132	43	3	1418	19371	132	0	0
20	16022	45	0	0	18083	44	0	0	16821	44	5	503	18147	135	0	0
21	16777	48	0	0	17480	48	0	0	17472	45	2	254	19500	126	0	0
22	16004	44	0	0	15771	46	0	0	16579	42	5	1809	17698	138	1	407
23	16725	43	0	0	19838	45	0	0	17367	44	2	241	18450	127	0	0
24	15466	45	0	0	16440	43	0	0	15479	43	2	857	17536	138	0	0
25	15711	49	0	0	19190	52	0	0	15911	47	6	2222	17104	142	1	404
26	17077	44	0	0	21300	48	0	0	17305	44	2	1078	18210	125	0	0
27	17916	47	0	0	21190	48	0	0	18568	44	3	1413	19031	137	0	0
28	17510	46	0	0	17779	51	0	0	17550	45	5	1433	20314	140	0	0
29	17342	43	0	0	18225	46	0	0	17232	41	3	1106	19545	131	1	115
30	16516	37	0	0	16893	38	0	0	17319	36	1	197	19378	130	0	0

Complete dominance Incomplete dominance

Although the ISBO minimally outperforms the hybrid framework for minimizing the number of the major setup times, it fails to avoid violations in the delivery dates and reports in average 1117 minutes total tardiness per problem instance. On the contrary, GA provides good results without any violations except one problem instance. The presented framework reports a complete dominance in terms of minimizing all objective values for solving twenty-three problem instances in comparison to the pure GA. These solutions are highlighted in blue in Table 1. Besides, partial dominance can be concluded for solving the rest of the considered problem instance as highlighted in grey in the same table. However, the outperformance of the GA for minimizing the number of major setups in these problems is very minimal. Besides, the proposed approach outperforms both heuristics and the GA in terms of the makespan in at least 93% of the considered problem instances. Generally, the proposed framework is able to dominate all approaches in minimizing at least three objective values and a minimal difference in the fourth objective value.

6. Conclusions and future research directions

In this paper, we proposed a hybrid scheduling framework to address scheduling problems in manufacturing environments adaptively. The presented proof of concept is evaluated for solving a multi objective HFS scheduling problem. The presented evaluation is conducted on thirty problem instances that are extracted from a real manufacturing environment in contrast to many contributions in the field of HFS scheduling [34]. The reported results increase the confidence in the stability and the robustness of the framework for delivering high-quality solutions taking into consideration various objective concerns of industrial environments. In addition, the framework delivers high-quality solutions with light execution time to adaptively react to the dynamic changes in the shop floor. Although, the optimization is carried out on a normal notebook with the following characteristics (CPU 4 x 2.6 GHz, RAM 8 GB), the average required computational effort by the hybrid approach for solving the problem instances is 5:57 minutes. Pure GA requires in average 50:12 minutes for solving the problems. This correspond to ten times more computational effort than the hybrid approach. Detailed results of the computational effort for solving the problems are presented in Figure 4.

Based on this result, we suggest further investigation of the potentials and minor limitations of hybrid solution strategies for solving different scheduling problems. Whether hybrid strategies outperform the robustness of metaheuristics techniques remains an open question that require further evaluation of such techniques for solving different problems.

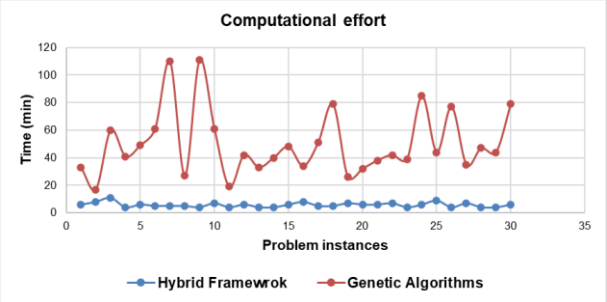

Figure 4. Required computational effort

References

[1] A. Sudiarso and A. W. Labib, "A fuzzy logic approach to an integrated maintenance/production scheduling algorithm", International Journal of Production Research, 40(13), 2002, pp. 3121–3138.

[2] Andersson, M., A.H.C. Ng, and H. Grimm, "Simulation Optimization for Industrial Scheduling Using Hybrid Genetic Representation", in Proceedings of the 40th Conference on Winter Simulation. 2008. Winter Simulation Conference.

[3] Aurich, P., A. Nahhas, T. Reggelin, and J. Tolujew, "Simulation-based Optimization for Solving a Hybrid Flow Shop Scheduling Problem", in Proceedings of the 2016 Winter Simulation Conference. 2016. IEEE Press: Piscataway, NJ, USA.

[4] Baker, K.R. and D. Trietsch, Principles of Sequencing and Scheduling, Wiley-Blackwell, Oxford, 2009.

[5] Barreto, L., A. Amaral, and T. Pereira, "Industry 4.0 implications in logistics: An overview", Procedia Manufacturing, 13, 2017, pp. 1245–1252.

[6] Bertel, S. and J.-C. Billaut, "A genetic algorithm for an industrial multiprocessor flow shop scheduling problem with recirculation", European Journal of Operational Research, 159(3), 2004, pp. 651–662.

[7] Brah, S.A. and J.L. Hunsucker, "Branch and bound algorithm for the flow shop with multiple processors", European Journal of Operational Research, 51(1), 1991, pp. 88–99.

[8] Glover, F., "Tabu search—part I", ORSA Journal on computing, 1(3), 1989, pp. 190–206.

[9] Glover, F. and G.A. Kochenberger, eds., Handbook of Metaheuristics, Springer Science & Business Media, 2003.

[10] Goldberg, D.E., Genetic algorithms in search, optimization, and machine learning, Addison-Wesley, Reading, Mass., Wokingham, 1989.

[11] Graham, R.L., E.L. Lawler, J.K. Lenstra, and Rinnooy Kan, A. H. G, "Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey", in Discrete Optimization II Proceedings of the Advanced Research Institute on Discrete Optimization and Systems

Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium co-sponsored by IBM Canada and SIAM Banff, Aha. and Vancouver, E.J. P.L. Hammer and B.H. Korte, Editors. 1979. Elsevier.
[12] Gupta, J.N.D., "Two-Stage, Hybrid Flowshop Scheduling Problem", The Journal of the Operational Research Society, 39(4), 1988, p. 359.
[13] Haouari, M., L. Hidri, and A. Gharbi, "Optimal Scheduling of a Two-stage Hybrid Flow Shop", Mathematical Methods of Operations Research, 64(1), 2006, pp. 107–124.
[14] Held, M. and R.M. Karp, "A Dynamic Programming Approach to Sequencing Problems", in Proceedings of the 1961 16th ACM National Meeting. 1961. ACM: New York, NY, USA.
[15] Hunsucker, J.L. and J.R. Shah, "Comparative performance analysis of priority rules in a constrained flow shop with multiple processors environment", European Journal of Operational Research, 72(1), 1994, pp. 102–114.
[16] Kacem, I., S. Hammadi, and P. Borne, "Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems", IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews), 32(1), 2002, pp. 1–13.
[17] Kirkpatrick, S., C.D. Gelatt, and M.P. Vecchi, "Optimization by simulated annealing", Science (New York, N.Y.), 220(4598), 1983, pp. 671–680.
[18] Kis, T. and E. Pesch, "A review of exact solution methods for the non-preemptive multiprocessor flowshop problem", European Journal of Operational Research, 164(3), 2005, pp. 592–608.
[19] Konak, A., D.W. Coit, and A.E. Smith, "Multi-objective optimization using genetic algorithms: A tutorial", Reliability Engineering & System Safety, 91(9), 2006, pp. 992–1007.
[20] Kut C. So, "Some Heuristics for Scheduling Jobs on Parallel Machines with Setups", Management Science, 36(4), 1990, pp. 467–475.
[21] Lang, S., T. Reggelin, F. Behrendt, and A. Nahhas, "Evolving Neural Networks to Solve a Two-Stage Hybrid Flow Shop Scheduling Problem with Family Setup Times", in Proceedings of the 53rd Hawaii International Conference on System Sciences. 2020.
[22] Lenstra, J.K., A. Rinnooy Kan, and P. Brucker, "Complexity of Machine Scheduling Problems", in Studies in Integer Programming, P.L. Hammer, E.L. Johnson, B.H. Korte and G.L. Nemhauser, Editors. 1977. Elsevier.
[23] Li, S., "A hybrid two-stage flowshop with part family, batch production, major and minor set-ups", European Journal of Operational Research, 102(1), 1997, pp. 142–156.
[24] Liao, C.-J., C.-T. Tseng, and P. Luarn, "A discrete version of particle swarm optimization for flowshop scheduling problems", Computers & Operations Research, 34(10), 2007, pp. 3099–3111.
[25] Miller, B.L. and D.E. Goldberg, "Genetic algorithms, tournament selection, and the effects of noise", Complex systems, 9(3), 1995, pp. 193–212.
[26] Mitchell, M. and J.H. Holland, "When will a genetic algorithm outperform hill-climbing?", 1993.
[27] Nahhas, A., P. Aurich, S. Bosse, T. Reggelin, and K. Turowski, "Metaheuristic and hybrid simulation-based optimization for solving scheduling problems with major and minor setup times", in 16th International Conference on Modeling and Applied Simulation, MAS 2017, Held at the International Multidisciplinary Modeling and Simulation Multiconference, I3M 2017, A. Bruzzone, A. Solis, M. Massei, F. De Felice, F. Longo, and C. Frydman, Editors.
[28] Nahhas, A., P. Aurich, T. Reggelin, and J. Tolujew, "Heuristic and Metaheuristic Simulation-Based Optimization for Solving a Hybrid Flow Shop Scheduling Problem", in The 15th International Conference on Modeling and Applied Simulation, A. G. Bruzzone, F. De Felice, C. Frydman, M. Massei, Y. Merkuryev, and A. Solis, Editors. 2016: RENDE (CS), ITALY.
[29] Nahhas, A., S. Lang, S. Bosse, and K. Turowski, "Toward Adaptive Manufacturing: Scheduling Problems in the Context of Industry 4.0", in 2018 Sixth International Conference on Enterprise Systems: ES 2018 : 1-2 October 2018, Limassol, Cyprus : proceedings, 2018 Sixth International Conference on Enterprise Systems (ES), Limassol, 1/10/2018 - 2/10/2018. 2018. IEEE: Piscataway, NJ.
[30] Oğuz, C. and M.F. Ercan, "A Genetic Algorithm for Hybrid Flow-shop Scheduling with Multiprocessor Tasks", Journal of Scheduling, 8(4), 2005, pp. 323–351.
[31] Pinedo, M.L., Scheduling: Theory, Algorithms, and Systems, Springer New York, 2012.
[32] Reischauer, G., "Industry 4.0 as policy-driven discourse to institutionalize innovation systems in manufacturing", Technological Forecasting and Social Change, 132, 2018, pp. 26–33.
[33] Ruiz, R. and C. Maroto, "A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility", European Journal of Operational Research, 169(3), 2006, pp. 781–800.
[34] Ruiz, R. and J.A. Vázquez-Rodríguez, "The hybrid flow shop scheduling problem", European Journal of Operational Research, 205(1), 2010, pp. 1–18.
[35] şerifoğlu, F.S. and G. Ulusoy, "Multiprocessor task scheduling in multistage hybrid flow-shops: a genetic algorithm approach", Journal of the Operational Research Society, 55(5), 2004, pp. 504–512.
[36] Voß, S., "The Two — Stage Hybrid — Flowshop Scheduling Problem with Sequence — Dependent Setup Times", in Operations Research in Production Planning and Control, G. Fandel, T. Gulledge, and A. Jones, Editors. 1993. Springer Berlin Heidelberg.
[37] Voudouris, C. and E. Tsang, "Guided Local Search", in Handbook of Metaheuristics, F. Glover and G.A. Kochenberger, Editors. 2003. Springer Science & Business Media.
[38] Wittrock, R.J., "Scheduling parallel machines with major and minor setup times", International Journal of Flexible Manufacturing Systems, 2(4), 1990, pp. 329–341.