CIRP Journal of Manufacturing Science and Technology 33 (2021) 100–114

Contents lists available at ScienceDirect

CIRP Journal of Manufacturing Science and Technology

journal homepage: www.elsevier.com/locate/cirpj

# Quantum algorithms for process parallel flexible job shop scheduling

Berend Denkena[a], Fritz Schinkel[b], Jonathan Pirnay[b], Sören Wilmsmeier[a,*]

[a] Institute of Production Engineering and Machine Tools (IFW), Leibniz University Hanover, Germany
[b] Fujitsu Technology Solutions GmbH, Germany

ARTICLE INFO

*Article history:*

Available online 23 March 2021

*Keywords:*
Production planning and control
Flexible job shop scheduling
Digital annealer
Process parallel optimization

ABSTRACT

Flexible Job Shop Scheduling is one of the most difficult optimization problems known. In addition, modern production planning and control strategies require continuous and process-parallel optimization of machine allocation and processing sequences. Therefore, this paper presents a new method for process parallel Flexible Job Shop Scheduling using the concept of quantum computing based optimization. A scientific benchmark and the application to a realistic use-case demonstrates the good performance and practicability of this new approach. A managerial insight shows how the approach for process parallel flexible job shop scheduling can be integrated in existing production planning and control IT-infrastructure.



## Introduction

Rising customers' expectations and technological developments have led to an increased complexity of manufacturing systems. Additionally, companies face stochastic disturbances and cyclic demands, which results in an unbalanced utilization of manufacturing capacity. Thus, effective and efficient production planning and control (PPC) have become a crucial advantage in competition. While work planning defines the required production technologies and strategies as well as the sequence of the production steps, production control is concerned with reoccurring activities in production, like order release or machine tool allocation, and short-term rescheduling due to unplanned machine breakdowns or deviations from planned times. However, the separation of work planning and production control has proven to be too rigid and inefficient in case of single-part and small-series production [1,49]. Integrated PPC enables shorter throughput times, transparent decision-making and a high level of ability to react in case of faults and unplanned events. Therefore, Denkena et al. [1] developed the approach of adaptive PPC (cf. Fig. 1), where alternative process chains for each order are generated totally automatically and PPC activities are completely integrated.

But, within this classical form of adaptive PPC an order perspective is presumed when calculating the priority value for each process chain. While the different process chains of an order are valuated, the routings of the other released orders are assumed to be fixed. This means that it is not taken into account that the choice of a process chain for an order can always affect the choice of production routes for all other orders. As a result, the best solution can be missed. Overall, a method is needed which optimizes the process parallel decision-making of the adaptive PPC holistically.

## State of the art

In order to counteract the lack of a production system perspective when selecting process chains, the approach of classical adaptive PPC was extended by a Genetic Algorithm [2]. Each individual forms a possible combination of work plans. By crossover, mutation and selection operations, a well suited work plan selection for all released orders can be determined in an acceptable time. The computing time for this pure routing optimization is still low enough to allow a process parallel use in shop floor production. With a maximum of 20 simultaneously released orders, 4 production levels and 19 machines in total, the computing time was always less than four minutes [3].

The holistically optimized allocation of work plans in combination with sequencing is known in research as the Flexible Job Shop Scheduling Problem (FJSP). The FJSP is an extension of the classical Job Shop Scheduling Problem (JSP), which allows an operation to be processed by some machines from a given set. The goal is to assign each operation to a machine, and to order the operations on the machines, such that the makespan of all operations is minimized [4]. The FJSP is considered one of the most difficult combinatorial optimization problems, and it was shown to be strongly NP-hard because the classical JSP has proven to be strongly NP-hard [5]. Because of this, heuristics are used for solving the problem.

* Corresponding author.
*E-mail address:* wilmsmeier@ifw.uni-hannover.de (S. Wilmsmeier).

https://doi.org/10.1016/j.cirpj.2021.03.006

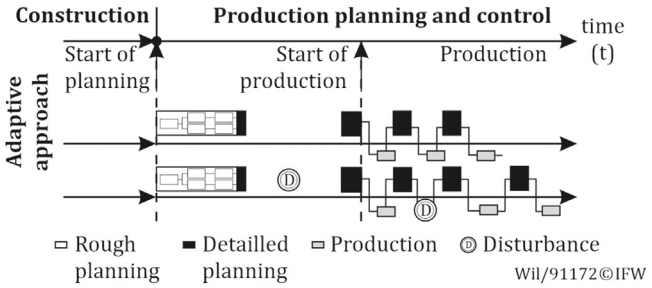


**Fig. 1.** Classical adaptive PPC (based on [1]).

As shown in Fig. 2, there is a variety of approaches to solving the FJSP. Heuristics are used both individually and in hybrid combinations. Genetic Algorithms, which are used for example in Pezzella et al. [6], Bagheri et al. [7], Wang et al. [8], Giovanni and Pezzella [9], Zhang et al. [10], Teekeng and Thammano [11], Alcan and Basligi [12], Chiang and Lin [13] and Driss et al. [14], are also a focal point of research. Furthermore, the approaches of Sun et al. [15] and Huang et al. [16] combine Genetic Algorithms with particle swarm optimization. Dalfard and Mohammadi [17] use Simulated Annealing to complement the Genetic Algorithm, Rossi and Boschi [18] combine the Genetic Algorithm with an Ant Colony approach. Both Li and Gao [19] and Zhang et al. [20] rely on a hybrid of Genetic Algorithm and Tabu Search. The use of Tabu Search forms the second larger cluster among the heuristics used.

In addition to Genetic Algorithms, Tabu Search, Ant Colony, Particle Swarm Optimization and Simulated Annealing, many other approaches can be found in research work on the FJSP. However, these usually use hybrid forms with the heuristics already mentioned (e. g., [21–31]). The approaches of Xing et al. [32] and Nouri et al. [33] are particularly noteworthy here. The first one uses simple rule based strategies for job assignment and sequencing. Nouri et al. [33] propose a hybridization of the two most frequently used metaheuristics (Genetic Algorithm and Tabu Search) within a holonic multiagent model.

The comparability of the different optimization approaches is partially given, since standardized test scenarios were already developed for example by Brandimarte [4], Kacem et al. [21] and Kacem et al. [22]. Subsequently, these have often been used by authors to benchmark their approach (cf. Table 1). The test scenarios differ with regard to the number of available machines (M), the orders to be scheduled (N) and the non-linear work plans underlying these orders.

In general, the use of the presented approaches was only intended for nonlinear PPC. A process parallel use, as it is necessary for adaptive PPC, was not in focus. This new field of application brings along a higher time criticality in the solution finding, because orders can only be passed on to the next workstation after a successful recalculation.

Depending on the size of the PPC problem, the computing time is documented for example in Xing et al. [32], Wang et al. [8], Li et al. [30], Zhang et al. [10], Nouri et al. [33] and Li and Gao [19]. As expected, the computing time usually increases significantly with the problem size. An exception is, for example, the calculation time for the MK-08 problem in Zhang et al. [10]. At 2.2 s, this is less than the solution times for the smaller problems MK-02 and MK-04 to MK-07. For the equally large problem MK-09 ($20 \times 10$) and for the problem instance seti5xxx ($15 \times 18$), significantly higher computation times of 30.2 s and 105.25 s are specified. It can therefore be assumed that the mechanism used by Zhang et al. [10] to form a starting population for the developed Genetic Algorithm happens to be ideally suited to the present PPC problem MK-08. Even the rule-based approach of Xing et al. [32] cannot show consistently low computing times that can be used for process parallel application. On the contrary, with 3.02–122.52 min, depending on the complexity of the optimization scenario, the times here are even significantly higher than the otherwise reported time range.

Basically, the scientific results suggest that, at the latest for problem sizes relevant to practice, a process-parallel application, as it becomes necessary with a novel combination with adaptive PPC, no longer seems realistic. Therefore, an approach is needed that provides results similar to or equal to those of the previous algorithms, but at the same time delivers reliably low computing times, especially for large problem instances. Ho et al. [24] present a first approach here, since the computation time of their CDR-PopGen algorithm still has a very low computation time of 0.042 s even for the largest MK problem instance ($20 \times 15$). With their approach, however, they do not achieve the best known solution for any MK problem (see Table 1).

In addition to the consideration of solution quality and computing time, a usable algorithm must ensure the import of optimization problems from and the export of solutions to existing planning and scheduling systems (e.g. Manufacturing Execution System, abbreviated MES) via generally available interfaces. This is the only way to avoid parallel structures and thus keep hurdles for the transfer into practical application low. This aspect is usually not considered in previous considerations.

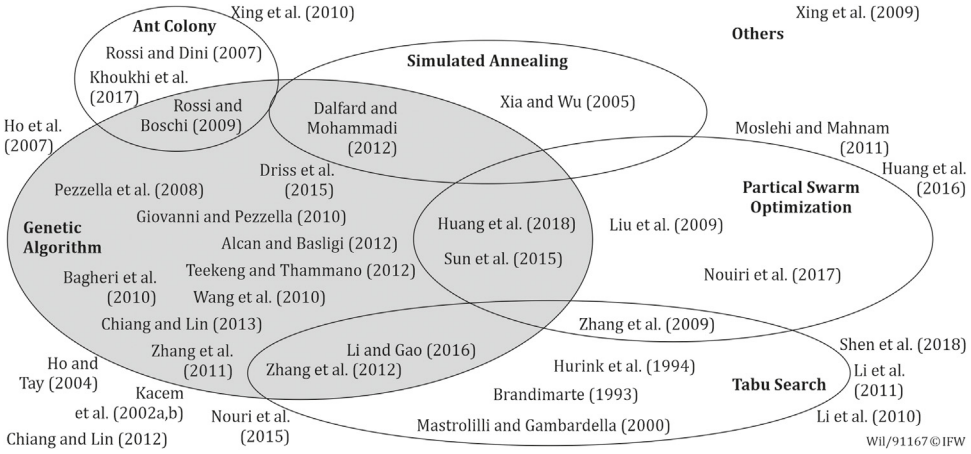


**Fig. 2.** Approaches to solving the FJSP [44–46,50–52].

**Table 1**
Comparison of solutions for the FJSP.

| | NxM | Brandimarte [4] | Mastrolilli and Gambardella (2000) [47] | Ho and Tay (2004) [23] | Ho et al. [24] | Pezella et al., [6] | Xing et al. [32] | Li et al. [25] | Bagheri et al. [7] | Wang et al. [8] | Xing et al. [26] | Zhang et al. [10] | Li et al. [30] | Teekeng and Thammano [11] | Chiang and Lin [13] | Nouri et al. [33] | Driss et al [14] | Huang et al. [29] | Li and Gao [19] | Nouiri et al. [48] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MK-01 | 10 × 6 | 42 | 40 | 41 | 42 | 40 | 42 | 40 | 40 | 40 | 39 | 40 | 40 | 40 | 40 | 40 | **37** | 40 | 40 | 40 |
| MK-02 | 10 × 6 | 32 | **26** | 29 | 30 | **26** | 28 | **26** | **26** | **26** | 29 | **26** | **26** | 27 | **26** | 27 | **26** | **26** | **26** | 27 |
| MK-03 | 15 × 8 | – | **204** | – | – | **204** | **204** | **204** | **204** | **204** | **204** | **204** | **204** | **204** | **204** | **204** | **204** | **204** | **204** | **204** |
| MK-04 | 15 × 8 | 81 | **60** | 67 | 68 | **60** | 68 | 61 | **60** | **60** | 65 | **60** | 62 | 62 | 61 | 64 | **60** | 61 | **60** | 64 |
| MK-05 | 15 × 4 | 186 | 173 | 176 | 179 | 173 | 177 | **172** | 173 | 173 | 173 | 173 | **172** | 178 | 173 | 173 | 173 | 173 | **172** | 173 |
| MK-06 | 10 × 15 | 86 | 58 | 68 | 69 | 63 | 75 | 65 | 63 | 60 | 67 | 58 | 65 | 78 | 65 | 65 | 67 | 62 | **57** | 65 |
| MK-07 | 20 × 5 | 157 | 144 | 148 | 153 | **139** | 150 | 140 | 140 | **139** | 144 | 145 | 140 | 147 | 140 | 144 | 148 | **139** | **139** | 144 |
| MK-08 | 20 × 10 | **523** | **523** | **523** | 527 | **523** | **523** | **523** | **523** | **523** | **523** | **523** | **523** | **523** | **523** | **523** | **523** | **523** | **523** | **523** |
| MK-09 | 20 × 10 | 369 | **307** | 328 | 326 | 311 | 311 | 310 | 312 | 310 | 311 | **307** | 310 | 341 | 311 | 311 | **307** | 310 | **307** | 311 |
| MK-10 | 20 × 15 | 296 | 198 | 231 | 243 | 212 | 227 | 214 | 214 | 214 | 229 | 199 | 214 | 252 | 225 | 222 | 212 | 214 | **197** | 222 |
| ∑ | **New best** | 9 | 4 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 2 | 0 |

## Concept of the Digital Annealer

In the past the solution of difficult optimization problems moved forward by innovative algorithms and the development of computer hardware. Indeed for the last 50 years progress in compute power followed Moore's law. Based on miniaturization of structures the hardware performance for von Neumann architectures doubled every 18 month. Physical laws limit this development when miniaturization arrives at atomic scale. Further improvements require new approaches and a revival of purpose-built machines is emerging [34].

Examples for promising hardware alternatives are neuromorphic computing delivering superior power efficiency to artificial intelligence applications or quantum computing to overcome the challenges of NP hard problems.

Combinatorial optimization for discrete decisions is the algorithmic background of many real world problems like cost reduction, risk mitigation, similarity search, task scheduling etc. These problems often turn out to be NP hard and economically reasonable solutions on classical hardware are out of reach. For such problems, quantum computers can be a solution. According to Lucas [35] many NP hard problems can be mapped to an energy minimization exercise formulated as an Ising model.

Quantum Annealing as a special concept of Quantum Computing is able to solve Ising model problems in polynomial time. Published by Kadowaki and Nishimori [36] as concept, Quantum Annealing was realized by D-Wave and commercialized since 2011 ([37]). In future generations, D-Wave could increase the number of bits and their connectivity and precision. D-Wave machines can solve Ising models but the required connectivity and precision in real world problems can only be reached by mapping the logical model bits to sets of physical qubits. This so called embedding process reduces the number of usable model bits by a factor of 30–50 compared to the available qubits. This limits the application of Quantum Annealing in operative scenarios today.

In many optimization applications the small compute effort and execution time is more important than the global optimality of the solution. For such applications, where the last percent of optimization target can be neglected, an approach on classical CMOS hardware is available. The Digital Annealer is a purpose-built machine by Fujitsu that optimizes Ising models in short processing times [38]. It can be applied to the same problem class as D-Wave Quantum Annealer. In its version 2 it can support fully connected high precision Ising models with 8192 bits.

The input for Quantum Annealing is a specific Ising model, which contains all information about the optimization problem. An Ising model is given as a binary search space $X = \{0,1\}^N$ for a natural dimension $N$ and an energy function $E : X \to \mathbb{R}$ from the search space to the real numbers that for each $\boldsymbol{x} = (x_1, \ldots, x_N) \in X$ is given by

$$E(x_1, \ldots, x_N) = \sum_{i=1}^{N} b_i x_i + \sum_{i=1}^{N} \sum_{j=i}^{N} w_{i,j} x_i x_j \tag{1}$$

for a real valued bias vector

$$\boldsymbol{b} = (b_1, \ldots, b_N) \in \mathbb{R}^N \tag{2}$$

and an upper triangular weight matrix

$$\boldsymbol{w} = (w_{i,j})_{\substack{i = 1,2,\ldots N \\ j = 1,2,\ldots N}} \in \mathbb{R}^{N^2}. \tag{3}$$

The quantum annealing searches for a solution $\boldsymbol{x_{min}} \in X$ with minimum energy, i.e.

$$\forall \boldsymbol{x} \in X : E(\boldsymbol{x}) \geq E(\boldsymbol{x_{min}}). \tag{4}$$

The polynomial $E$ of quadratic order in binary variables $x_i$ is called Quadratic Unconstrained Binary Optimization (QUBO). The Digital Annealer gets such a QUBO as input and therefore supports exactly the same problems and problem formulation as Quantum Annealers do. It can handle QUBOs defined on up to 8192 bits and the bias and weight values can be represented in 16–64 bit precision. This allows much bigger models than possible on quantum technologies today.

On the Digital Annealer, the search for optimized solutions is a Markov Chain Monte Carlo process. A step from a current state $\boldsymbol{x} \in X$ to a neighbor state $\boldsymbol{x}' \in X$ differing in one bit from the current state is applied with a probability depending on the energies of the two states where lower energies are preferred. The base concept is a simulated annealing approach [39,40], where the acceptance of a step from one state to another additionally depends on a start temperature $\tau$ which is gradually decreased during the process. The start and end temperature and the number of steps for the random walk are figured out problem specific and are typically not modified for every data set. Hence the calculation time can be kept constant and does not grow exponentially with the problem size. The Digital Annealer has multiple hardware and algorithmic improvements compared to simulated annealing, and it can provide very good optimization results in sub second runtime:

1) Parallel trial steps: Every state in the search space has $N$ neighbors in Hamming distance of one bit. The random experiments for accepting a neighbor as next position candidate are done in parallel and then one of the candidates is selected. This shortens the waiting time due to unaccepted steps in local minima by up to factor of $N$.
2) Fast escape from local minimum: When the annealing process is in low temperature even with parallel attempts for all neighbors no candidate might be produced. In that case the next attempt is calculated from a higher "borrowed" start energy to increase the acceptance probability. With every miss of candidates after a trial phase this energy is increased so that latest after few energy increments a step acceptance becomes sure and the local minimum can be left behind. This helps the search in the later phases at cold temperature to do much more steps and find better solutions in shorter time.
3) Parallel execution: The Digital Annealer hardware provides not only enough arithmetic logical units for parallel step attempts on all bits but 16 times more units. Therefore, it can run 16 independent annealing processes in parallel in the hardware. This increases the quality of the best solution.
4) Parallel tempering replica exchange: Aside from annealing, the Digital Annealer supports parallel tempering as a second optimization approach. Annealing processes at different constant temperatures are executed in parallel. After a given number of steps, the reached states are shuffled between neighboring temperatures randomly where higher energy preferably goes to higher temperature to give the worse state higher probability to search around while the better state goes into colder converging and conserving annealing conditions.

A detailed description of the Digital Annealer together with some benchmarking results can be found in Matsubara et al. [41]. In contrast to ideal quantum annealers the algorithmic approach of the Digital Annealer cannot guarantee an optimum solution in every case. Combinatorial problems with few valuable solutions or even only one result of interest are less suited for the Digital Annealer; factorization of large integers as a possibility to break encryptions is a popular example in that direction. On the other hand, if the set of admissible solutions is dense in the binary search space and the QUBO allows several near optimum solutions, then the problem is typically well suited for the Digital Annealer.

Current quantum annealers are limited in the representation of densely connected Ising models, i.e. problems where the weight matrix of Eq. (3) features a high number of off-diagonal nonzero entries [42]. This limitation of physical qubit coupling does not apply to simulated annealing approaches. The quantum inspired algorithmic improvements and hardware accelerations of simulated annealing gives the Digital Annealer real time capabilities for many industrial optimization problems. Thus, the concept of the Digital Annealer fulfils in principle the requirements formulated in chapter 2 for a solution approach for process-parallel, holistically optimized work plan selection within the framework of adaptive PPC. To apply the Digital Annealer to the FJSP, however, a suitable Ising model is still required. Therefore, in the next chapter different transformations of the FJSP into Ising models are demonstrated.

## QUBO-formulation for process parallel flexible job shop scheduling

In this chapter a formulation of the FJSP that is amenable to quantum annealers is constructed, i.e. a formulation of a FJSP instance as a quadratic unconstrained optimization problem is needed. The following QUBO-formulation is based on the work of Venturelli et al. [43] and extends it with a penalty term that favors schedules with shorter makespans. Additionally an approach of how to efficiently estimate a discretized timeline from original problem instances in continuous time is presented.

### *Problem definition*

A (continuous) FJSP instance consists of a set of $n$ jobs $J = \{j_1, \ldots, j_n\}$ which need to be scheduled on a set of machines $M = \{m_1, \ldots m_y\}$. For each job $j \in J$ there is a set of operations $O_j$ with $n_j := |O_j|$, all of which need to be executed in a defined order. That is, let $O := \bigcup_{j \in J} O_j$ be the union of all operations, then there is a unique order mapping

$$c : O \to \mathbb{N}, \tag{5}$$

where for each $j \in J$ the restriction $c|_{O_j}$ maps $O_j$ bijectively onto the set $\{1, \ldots, n_j\}$. It can be assumed that for any job $j \in J$ and $o \in O_j$ the position of the operation $o$ in the execution order of job $j$ is given by $c(o)$.

Depending on the problem statement, an operation has multiple machines on which it can potentially be executed. That is, for any $j \in J$ and $o \in O_j$ there is a non-empty subset $M_o \subseteq M$ of machines that are *capable of performing operation* $o$. Furthermore for any $m \in M$ and $o \in O_j$ *the duration of operation* $o$ *on machine* $m$ is denoted by $d_{o,m} \in \mathbb{R}_{\geq 0}$. In particular it is assumed that $d_{o,m} = 0$ if and only if $m \notin M_o$.

Additionally, as bigger problems will later be split up into smaller instances, it is assumed that for each $m \in M$ there is a *machine availability time* $a_m \in \mathbb{R}_{\geq 0}$, which indicates the earliest possible time at which an operation can be started on machine $m$. Analogously a *job availability time* $a_j \in \mathbb{R}_{\geq 0}$ is assigned to each job $j \in J$, indicating the earliest possible time at which the first operation of a job $j$ is allowed to start. Hence an operation of a job $j$ can at the earliest start on some machine $m$ at time $\max\{a_j, a_m\}$.

A *schedule* $V$ for a FJSP $J, O, M$ as above is defined as a set of triples $(o, m, t) \in O \times M \times \mathbb{R}_{\geq 0}$ (indicating the starting time $t$ of an operation $o$ on some machine $m$) which satisfies:

1) For all $o \in O$, there is at most one $m \in M$ and at most one $t \in \mathbb{R}_{\geq 0}$ with $(o, m, t) \in V$.
2) For all $(o, m, t) \in V$ it is satisfied $d_{o,m} > 0$.

3) Let $(o,m,t),(o',m',t') \in V$ with $o,o' \in O_j$ for some $j \in J$. If $c(o) < c(o')$, then $t < t'$.

A *feasible schedule* $V$ is a schedule where the further conditions hold:

4) For all $o \in O$, there is exactly one $m \in M$ and exactly one $t \in \mathbb{R}_{\geq 0}$ with $(o,m,t) \in V$.
5) Let $(o,m,t) \in V$ with $o \in O_j$ for some $j \in J$. Then $\max\{a_j, a_m\} \leq t$.
6) Let $(o,m,t),(o',m',t') \in V$ with $o,o' \in O_j$ for some $j \in J$. If $c(o) < c(o')$, then $t + d_{o,m} \leq t'$.
7) Let $m \in M$. For all $o \neq o' \in O$ and $t,\ t' \in \mathbb{R}_{\geq 0}$ with $(o,m,t),(o',m,t') \in V$ the time $t'$ does not lie in the half open interval $[t, t + d_{o,m}$.

A feasible schedule $V$ is *squeezed*, if for any $(o,m,t) \in V$ with $o \in O_j$ for some $j \in J$ at least one of the following is true:

8) The time $t$ satisfies $t = \max\{a_j,\ a_m\}$.
9) For $o' \in O_j$ with $c(o) = c(o') + 1$, there is $m' \in M$ such that $\left(o', m', t - d_{o',m'}\right) \in V$.
10) There exists $o' \in O$ such that $\left(o', m, t - d_{o',m}\right) \in V$.

A squeezed schedule is uniquely given by the order of operations on each machine. Indeed, it is easy to see that by shifting the starting times accordingly, for any schedule $V$ satisfying condition 4) it can be passed to a unique squeezed schedule $V_{sq}$ with the same order of operations on each machine, i.e. for all $m \in M$ and $o,o' \in O$ applies

$$\exists t < t' \in \mathbb{R}_{\geq 0}\ \ s.t.\ \ (o,m,t),\left(o',m,t'\right) \in V \Longleftrightarrow \exists t < t' \in \mathbb{R}_{\geq 0}\ \ s.t.\ \ (o,m,t),\left(o',m,t'\right) \in V_{sq}. \tag{6}$$

Finally the *makespan* of a schedule $V$ is defined as

$$\text{makespan}(V) := \max_{(o,m,t) \in V} \{t + d_{o,m}\}, \tag{7}$$

and the objective of a FJSP $J$, $O$, $M$ as above is to find

$$\min_{\text{feasible schedule } V} \text{makespan}(V). \tag{8}$$

Note that for any feasible schedule $V$ applies $\text{makespan}(V_{sq}) \leq \text{makespan}(V)$.

For the QUBO formulation only discrete FJSP instances are considered. In later sections it is suggested how to discretize continuous FJSP instances. A discrete FJSP instance is given by jobs $J$, operations $O_j$, machines $M$ and a unique order map $c : O \longrightarrow \mathbb{N}$ exactly as above. Additionally, there is a discrete timeline $T' := \{0,\ldots,T-1\}$ for a number of time ticks $T \in \mathbb{N}$. The duration of an operation $o \in O$ on a machine $m \in M$ is given as a number of time ticks by $d_{o,m} \in \mathbb{N}$. As above for the job and machine availability times applies $a_j, a_m \in \mathbb{N}$.

The definition of a discrete (feasible, squeezed) schedule $V$ and its makespan works exactly as above, however in this case $V \subseteq O \times M \times T'$. For clarity the conditions of a discrete *feasible schedule* $V$ are given:

4) For all $o \in O$, there is exactly one $m \in M$ and exactly one $t \in T'$ with $(o,m,t) \in V$.
5) Let $(o,m,t) \in V$ with $o \in O_j$ for some $j \in J$. Then $\max\{a_j, a_m\} \leq t < T - d_{o,m}$.
6) Let $(o,m,t),(o',m',t') \in V$ with $o,o' \in O_j$ for some $j \in J$. If $c(o) < c(o')$, then $t + d_{o,m} \leq t'$.
7) Let $m \in M$. For all $o \neq o' \in O$ and $t,\ t' \in T'$ with $(o,m,t),(o',m,t') \in V$ applies $t' \notin \{t, \ldots t + d_{o,m} - 1\}$.

As in the continuous case, for any discrete schedule $V$ satisfying condition 4) it can be passed to a squeezed schedule $V_{sq}$ with the same order of operations on each machine. However, note that it might be necessary to move to an extended timeline $T'$ in the problem instance.

*QUBO setup*

In the following, let a discrete problem instance be given by jobs $J$, machines $M$ and operations $O$ as well as a timeline $T' := \{0,\ldots,T-1\}$ for some $T \in \mathbb{N}$ as above.

In this setup, a set of binary variables which corresponds to all the possible discrete starting times of the operations is assigned, i.e. binary variables

$$(x_{o,m,t})_{o \in O, m \in M, t \in T'} \tag{9}$$

are defined, where

$$x_{o,m,t} = \begin{cases} 1: & \text{operation } o \text{ starts on machine } m \text{ at time } t, \\ 0: & \text{otherwise.} \end{cases}.$$

In the following, a binary polynomial $H$ is set up in the variables $(x_{o,m,t})$ which is associated with the problem instance: First for the different necessary constraints in FJSP are accounted by defining binary polynomials $H_0$, $H_1$, $H_2$. Then a binary polynomial $H_3$ is defined that serves as a cost function which favors schedules with shorter makespans. Afterwards the optimization objective $H$ can be set as

$$H := \ \alpha H_0 + \beta H_1 + \gamma H_2 + \delta H_3 \tag{11}$$

with scalar weights $\alpha,\ \beta,\ \gamma,\ \delta \in \mathbb{R}_{>0}$.

*Polynomials for necessary constraints*

The optimization problem is limited by three constraints. Firstly, each operation must be started once and only once on some capable machine:

$$H_0 := \sum_{o \in O} \left(1 - \sum_{m \in M_o} \sum_{t=0}^{T-1} x_{o,m,t}\right)^2. \tag{12}$$

Obviously, this rule is fulfilled for a binary configuration $(x_{o,m,t})$ as above if and only if $H_0((x_{o,m,t})) = 0$.

Secondly, for every job $j \in J$, its operations $O_j$ must be executed in the order given by

$$c|_{o_j} : O_j \to \{1,\ldots,n_j\}. \tag{13}$$

For any $j \in J$ the set $V_j$ of violating index combinations for job $j$ is defined as

$$V_j = \left\{ \left((o,m,t),\left(o',m',t'\right)\right) \in \left(O_j \times M \times T'\right)^2 | m \in M_o \wedge m' \in M_{o'} \wedge \left(c(o) < c\left(o'\right)\right) \wedge \left(t' < t + d_{o,m}\right) \right\}$$

thus

$$H_1 := \sum_{j \in J} \sum_{((o,m,t),(o',m',t')) \in V_j} x_{o,m,t} x_{o',m',t'}. \tag{14}$$

Again, the rule described above is fulfilled for any binary configuration $(x_{o,m,t})$ if and only if $H_1((x_{o,m,t})) = 0$.

In addition, only one operation can be executed simultaneously on a single machine at a certain point in time. Analogously to above, for any $m \in M$ a set $W_m$ of violating index combinations for machine $m$ is defined as

$$W_m = \left\{ \left( (o,m,t), \left(o',m,t'\right) \right) \in \left( O \times M \times T' \right)^2 | t \le t' < t + d_{o,m} \right\},$$

thus

$$H_2 := \sum_{m \in M} \sum_{\left((o,m,t),\left(o',m,t'\right)\right) \in W_m} x_{o,m,t} x_{o',m,t'}. \tag{15}$$

Again, for any binary configuration $(x_{o,m,t})$ as above no machine is double-occupied if and only if $H_2((x_{o,m,t})) = 0$.

*Predecessors and successors*

Before the binary polynomial which serves as cost function for longer makespans is defined, the notion of predecessor and successor times is introduced: Let $j \in J$ and $o \in O_j$. Depending on $c(o)$, in a feasible schedule the operation $o$ cannot start arbitrarily early or late, since there has to be a minimum number of time ticks before (resp. after) the finishing time of $o$ to execute all operations $o' \in O_j$ with $c(o') < c(o)$ (resp. with $c(o') > c(o)$).

For any $o \in O$ denote by $d_{o,\min} = \min_{m \in M_o} \{d_{o,m}\}$ the minimum time in which an operation can in principle be executed.

For $j \in J$, $o \in O_j$ and $m \in M_o$, the minimum successor time $S_{j,o,m}$ for operation $o$ on machine $m$ is given by

$$S_{j,o,m} := d_{o,m} + \sum_{o' \in O_j,\ c(o') > c(o)} d_{o',\min}. \tag{16}$$

The *minimum predecessor time* $P_{j,o,m}$ for a job $j$ and an operation $o \in O_j$ on machine $m$ is given recursively along its execution order $c(o)$. For $o$ with $c(o) = 1$ set

$$P_{j,o,m} := \max\{a_m, a_j\}. \tag{17}$$

Let $P_{j,o',m}$ be already defined for some $o' \in O_j$ and let $o \in O_j$ with $c(o) = c(o') + 1$, then set

$$z_{j,o'} := \min_{m \in M_o} \left\{ P_{j,o',m} + d_{o',m} \right\}, \tag{18}$$

and recursively

$$P_{j,o,m} := \max\left\{ a_m, z_{j,o'} \right\} \tag{19}$$

*Makespan objective $H_3$*

The polynomials $H_0$, $H_1$, $H_2$ only consist of combinatorial constraints, however the optimization target is to bring the machine activities to early time ticks. With the notion of predecessor and successor times, a binary polynomial can be defined that penalizes late starting times of operations that could in theory have already started.

Let $j \in J$, $o \in O_j$, $m \in M_o$ and let $\Gamma_{j,o,m} = T - S_{j,o,m} - P_{j,o,m}$. If $\Gamma_{j,o,m} \le 0$, then the timeline $T' = \{0, \dots, T-1\}$ is too small and the operation $o$ has no possibility to start on machine $m$. Otherwise, for any time tick $t \in T'$ with

$$P_{j,o,m} \le t < T - S_{j,o,m} \tag{20}$$

a scalar penalty weight is defined

$$p_{o,m,t} := \frac{t + d_{o,m} - P_{j,o,m}}{\Gamma_{j,o,m}} \in \left[ \frac{d_{o,m}}{\Gamma_{j,o,m}}, 1 + \frac{d_{o,m}}{\Gamma_{j,o,m}} \right. . \tag{21}$$

and then

$$H_3 := \sum_{j \in J} \sum_{o \in O_j} \sum_{m \in M_o} \sum_{t = P_{j,o,m}}^{T - S_{j,o,m} - 1} p_{o,m,t} x_{o,m,t}. \tag{22}$$

All in all, this results in the complete objective binary polynomial according to Eq. (11). It is only necessary to define penalty weights for time ticks $t$ as in Eq. (20), because all further binary variables can be pruned, as described in the next section.

The optimization objective $H$ itself is unconstrained: the necessary constraints of the scheduling problem are encoded within the penalties $H_0, H_1, H_2$. In the annealing process these need to be driven to zero in order to obtain a feasible schedule. Depending on the problem instance it thus makes sense to choose $\alpha, \beta, \gamma \gg \delta$.

In general, due to the impact of $H_1$ and $H_2$, the objective polynomial $H$ will possess a high number of nonzero quadratic coefficients. Additionally, finding the global minimum of the FJSP is often unnecessary for practical industrial applications, whereas obtaining a near optimum solution is sufficient. This makes the optimization of the polynomial $H$ suitable for a simulated annealing approach with the Digital Annealer, as outlined at the end of chapter 3.

*Variable pruning*

The number of binary variables $(x_{o,m,t})_{o \in O, m \in M, t \in T'}$ equals $|O| \cdot |M| \cdot T$, however, the discussion above shows that the number of variables which is needed can be significantly reduced:

1) Not every machine can perform every operation, hence it can be stated that $x_{o,m,t} = 0$ for all $t \in T'$, $o \in O$, $m \in M$ with $m \notin M_o$.
2) As discussed, for an operation there needs to be time for predecessor and successor operations, hence for all $o \in O$ and $m \in M$ applies $x_{o,m,t} = 0$ for all time ticks $t \in T'$ with $0 \le t < P_{j,o,m}$ or $t \ge T - S_{j,o,m}$.

*Discretization of problem instances with real makespans*

In practice for a given continuous problem instance $J, O, M$ with real makespans $d_{o,m} \in \mathbb{R}_{\ge 0}$, a suitable discrete timeline $T'$ is not known in advance. In order to find a fitting discretization of the problem instance first a real candidate for a timeline $\Theta \in \mathbb{R}_{>0}$ in which all operations can be executed needs to be estimated. The timeline $\Theta$ then needs to be discretized by defining the number of equidistant time ticks $T \in \mathbb{N}$, from which the length of one timestep $l := \frac{\Theta}{T-1}$ can be computed. However, this poses several difficulties:

1) In a discretization, given a real length $l \in \mathbb{R}_{>0}$ of one timestep, the execution of some operation $o \in O$ on a machine $m \in M$ takes $\frac{d_{o,m}}{l}$ timesteps, which in general will not be an integer. Applying some rounding operator leads to inaccuracies in the representation of the original problem.
2) As the number of usable binary variables on current quantum annealers is severely limited (on the Digital Annealer it is limited to 8192 at the time of writing), the number of timesteps

$T$ should be chosen in a way that already respects the amount of binary variables which will be pruned as described in chapter 4.6.

In the following two sections these two considerations are addressed. In section 4.7.3 a two-step process to obtain a suitable discretization of a continuous problem instance is proposed.

*Discretization of execution times*

For a continuous problem instance $J$, $O$, $M$, suppose some real candidate for a timeline $\Theta \in \mathbb{R}_{>0}$ as well as a desired number of timesteps $T \in \mathbb{N}$ for a discretization is given. Formally, $\Theta$ is any positive real number for which the set of feasible schedules

$$\{V | V \text{ is } \quad \text{makespan}(V) \leq \Theta\} \tag{23}$$

is nonempty. Using the length of one time step $l := \frac{\Theta}{T-1}$, it is possible to pass to a discrete problem instance with timeline $T' := \{0, \ldots, T-1\}$ and integer durations $\tilde{d}_{o,m} := g\left(\frac{d_{o,m}}{l}\right)$, where g denotes some integer approximation such as flooring, ceiling or rounding to the nearest integer. Obtain discrete availability times $\tilde{a}_j$ and $\tilde{a}_m$ in the same way.

Now given a squeezed schedule $\tilde{V} \subseteq O \times M \times T'$ for the discretized problem, it can be passed to a continuous schedule $V$ for the original problem using the map

$$\tilde{V} \to O \times M \times \mathbb{R}_{\geq 0}, \quad (o, m, t) \mapsto (o, m, t \cdot l). \tag{24}$$

Even though it can always be passed from $V$ to $V_{sq}$, the schedule $V$ cannot be expected to be squeezed: if $g$ is the ceiling function, $V$ will be feasible, however will most likely have gaps between the operations. If $g$ is the flooring function, $V$ cannot be expected to be feasible, as there might be overlapping operations. Any integer approximation apparently cannot capture all subtleties in the real durations $d_{o,m}$, however, depending on $l$ they blur eminently for flooring and ceiling. In particular, discrete durations will be approximated in the following by basic rounding, where durations $< 1$ will always be rounded up to 1, in order to not include durations of 0 on capable machines. This means that for a given $l$, the map

$$g : \mathbb{R}_{\geq 0} \to \mathbb{N}, \quad d \mapsto \begin{cases} 0, & \text{if } d = 0, \\ 1, & \text{if } 0 < \frac{d}{l} < 0.5, \\ \left\lfloor \frac{d}{l} + 0.5 \right\rfloor & \text{otherwise.} \end{cases} \tag{25}$$

is used.

*Choosing a timeline T' given Θ*

In this section the problem of choosing the number of time ticks $T$ for a suitable discretization of a real makespan candidate $\Theta$ is addressed, while at the same time respecting some maximum number of binary variables as well as the number of variables which can be pruned as described in chapter 4.6.

The same notation as in section 4.7.1 above is used.

Let $J$, $O$, $M$ be a continuous problem instance, and suppose a real candidate for a timeline $\Theta \in \mathbb{R}_{>0}$ is given. Denote by $Q$ a *predefined maximum number of possible binary variables* that can be used. Let

$$U = \left| \{(o, m) \in O \times M | d_{o,m} > 0\} \right| \tag{26}$$

be the number of all (operation, capable machine)-pairs.

Furthermore given any $T \in \mathbb{N}_{>0}$, it is possible to pass on to a discrete problem instance as described in the previous section and $F(T)$ is written for the number of unused predecessor/successor variables as in chapter 4.6, depending on $T$. Thus for the discretization of the continuous problem $T$ should be chosen such that

$$T = \max\left\{\tilde{T} \in \mathbb{N}_{>0} | U \cdot \tilde{T} - F\left(\tilde{T}\right) \leq Q\right\}. \tag{27}$$

Let $\tilde{T} \in \mathbb{N}_{>0}$. In the continuous problem instance for each job $j \in J$, operation $o \in O_j$ and machine $m \in M$ *predecessor times* $P_{j,o,m}$ and *successor times* $S_{j,o,m}$ can be defined in the exact same way as in chapter 4.4 above (with the only exception that the durations $d_{o,m}$ lie in $\mathbb{R}$). Using that any duration $d \in \mathbb{R}$ takes $g\left(\frac{d}{l}\right)$ time steps of length $l := \frac{\Theta}{\tilde{T}-1}$, and by omitting the rounding operation $g$, an estimate

$$\begin{aligned} F\left(\tilde{T}\right) &\sim \sum_{j \in J, o \in O_j, m \in M_o} \left( \frac{P_{j,o,m} \cdot \left(\tilde{T} - 1\right)}{\Theta} + \frac{S_{j,o,m} \cdot \left(\tilde{T} - 1\right)}{\Theta} \right) \\ &= \frac{\tilde{T} - 1}{\Theta} \cdot \left( \sum_{j \in J, o \in O_j, m \in M_o} P_{j,o,m} + S_{j,o,m} \right) \end{aligned} \tag{28}$$

is obtained.

Hence using Eq. (27) and denoting

$$G := \sum_{j \in J, o \in O_j, m \in M_o} P_{j,o,m} + S_{j,o,m}, \tag{29}$$

an initial approximation of $T$ is obtained via

$$T \sim \left\lfloor \frac{Q - \frac{G}{\Theta}}{U - \frac{G}{\Theta}} \right\rfloor. \tag{30}$$

Using this estimate of $T$, a discretization of the given problem instance can be computed and hence an exact value of the number of predecessor/successor variables $F(T)$, as well as the number of used binary variables $\tilde{Q} = U \cdot T - F(T)$. Depending on the difference $\tilde{Q} - Q$, this estimate of $T$ can then be even further refined by iteratively decreasing respectively increasing $T$ and recomputing $\tilde{Q}$ until $T$ satisfies Eq. (27).

*Two-step process for discretization*

In order to choose a suitable discretization of a continuous problem instance $J$, $O$, $M$, it is proceeded as follows:

- A fast round-robin scheduling approach is used to find a squeezed schedule to the problem instance. This approach simply takes all jobs permuted arbitrarily and schedules the first operation of each job one after another, always choosing for each operation the most convenient free slot on the machines. Then the second operation of each job is scheduled and so on. This is repeated for $q$ times to obtain a set of $q$ squeezed schedules. Of these, let $V$ be the schedule with the minimum makespan, and denote its makespan by $\Theta'$.
- Given $\Theta'$, compute number of timesteps $T$ and a discretization as described in section 4.7.1 and 4.7.2 above.
- From the schedule $V$, using the same order of operations on each machine, pass to a discrete squeezed schedule $\tilde{V}$ for the discretized problem instance.
- Compute

$$\Theta := \varepsilon \cdot \text{makespan}\left(\tilde{V}\right) \cdot \frac{\Theta'}{T-1}, \tag{31}$$

for some $\varepsilon \geq 1$. Use $\Theta$ as an updated makespan candidate to arrive at an updated discretization, again using the approach described above in section 4.7.1 and 4.7.2.

Here $\varepsilon \geq 1$ serves as a security scaling factor to give the annealing process more freedom in the choice of timesteps when scheduling operations.

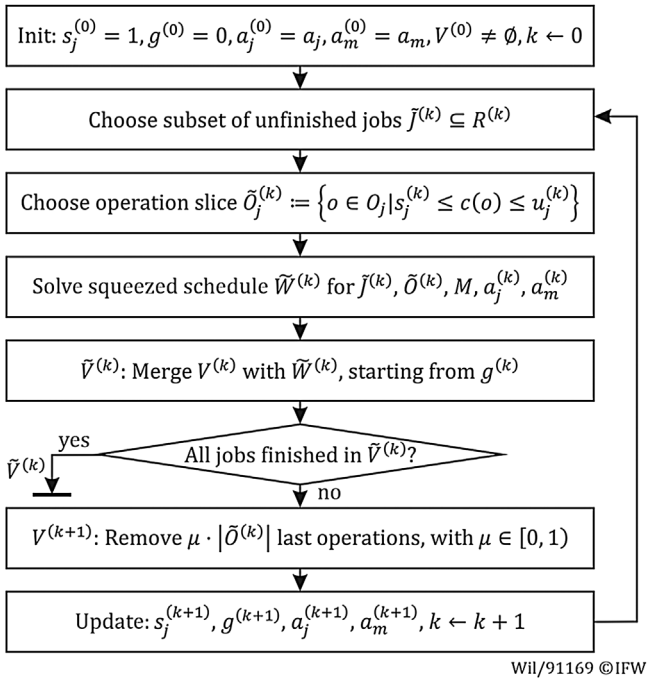


**Fig. 3.** Heuristic for solving large scale problems.

*Iteratively solving large-scale problems*

With a limited number of model bits for a growing number of jobs, operations and machines, the number of time ticks that can be used in a discretization of the continuous timeline shrinks. For example, the Digital Annealer currently has a limit of 8192 bits - depending on the problem instance, needing to fit around two hundred operations becomes too coarse-grained and no longer yields good solutions. The number of qubits on usable quantum annealers at the current state of technology is even lower.

Thus, in order to solve larger problem instances, these need to be split into smaller subproblems of which the solutions in turn are combined into a full solution of the original large problem. On a high level, the proposed iterative approach is described below and summarized in Fig. 3.

A job $j \in J$ is defined to be *unfinished* in some (continuous or discrete) schedule $V$, if there is $o \in O_j$ such that $o \neq o'$ for all $(o', m, t) \in V$. Analogously a job $j \in J$ is *finished* in a schedule $V$, if there exists $(o, m, t) \in V$ for all $o \in O_j$.

Given as usual a continuous problem instance $J$, $O$, $M$ with durations $d_{o,m} \in \mathbb{R}_{\geq 0}$ and job availability times $a_j = 0$ for all $j \in J$, and machine availability times $a_m = 0$ for all $m \in M$ and order map $c : O \mapsto \mathbb{N}$, the proposed iterative approach is as follows:

1) Let $\mu \in [0, 1$ be a predefined *ratio of operations to remove* after each iteration. Furthermore let $V^{(0)} := \emptyset$ be an empty *global schedule*. For all $j \in J$ let $s_j^{(0)} := 1$ be the *starting operation index*, as well as a *global starting time* $g^{(0)} := 0$. Define availability times $a_j^{(0)} := a_j$ for each $j \in J$ and $a_m^{(0)} := a_m$ for each $m \in M$. Set $k \leftarrow 0$.
2) Let $R^{(k)} \subseteq J$ be the subset of unfinished jobs in $V^{(k)}$. Choose an 'appropriate' subset $\tilde{J}^{(k)} \subseteq R^{(k)}$ of unfinished jobs. For each $j \in \tilde{J}^{(k)}$, a chain of operations $\tilde{O}_j^{(k)} \subseteq O_j$ is taken from the beginning of the remaining operations in $O_j$. That is, choose $u_j^{(k)} \in \left\{s_j^{(k)}, \ldots, n_j\right\}$ and set $\tilde{O}_j^{(k)} := \left\{o \in O_j | s_j^{(k)} \leq c(o) \leq u_j^{(k)}\right\}$, as well as $\tilde{O}^{(k)} := \bigcup_{j \in J}^{(k)} \tilde{O}_j^{(k)}$. Obtain an adjusted order function $c^{(k)} : \tilde{O}^{(k)} \to \mathbb{N}$, with $c^{(k)}(o) = c(o) - s_j^{(k)} + 1$ for $o \in \tilde{O}_j^{(k)}$ for some $j \in \tilde{J}^{(k)}$.
3) Solve the smaller subproblem consisting of $\tilde{J}^{(k)}$, $\tilde{O}^{(k)}$, $M$ with availability times $a_j^{(k)}$, $a_m^{(k)}$ and order map $c^{(k)}$ as described, and obtain a squeezed schedule $\widetilde{W}^{(k)}$ for the subproblem $\tilde{J}^{(k)}$, $\tilde{O}^{(k)}$, $M$. Merge the current global schedule $V^{(k)}$ with $\widetilde{W}^{(k)}$, i.e. set

$$\widetilde{V}^{(k)} := V^{(k)} \cup \left\{\left(o, m, g^{(k)}\right) | \left(o, m, t - g^{(k)}\right) \in \widetilde{W}^{(k)}\right\}. \tag{32}$$

If all jobs $j \in J$ are finished in $\widetilde{V}^{(k)}$, it is terminated with the squeezed schedule $V^{(k+1)} := \widetilde{V}^{(k)}$ for the initial problem $J$, $O$, $M$.

4) Otherwise, given $\mu$, a tail of the appended operations in $\widetilde{W}^{(k)}$ is cut off. That is, let $t^{(k)} \leq \text{makespan}\left(\widetilde{V}^{(k)}\right)$ be maximal such that

$$\left|\left\{(o, m, t) \in \widetilde{W}^{(k)} | t + d_{o,m} > t^{(k)}\right\}\right| \geq \mu \cdot \left|\widetilde{O}^{(k)}\right|. \tag{33}$$

Then set as next iterate for the current global schedule

$$V^{(k+1)} := V^{(k)} \cup \left\{(o, m, t) | \left(o, m, t - g^{(k)}\right) \in \widetilde{W}^{(k)} \wedge t - g^{(k)} + d_{o,m} \leq t^{(k)}\right\} \tag{34}$$

5) For each $m \in M$ let

$$f_m^{(k)} := \max\left\{t + d_{o,m} | (o, m, t) \in V^{(k+1)}\right\} \tag{35}$$

the finishing time of each machine, respectively $f_m^{(k)} := 0$ if the set on the right hand side is empty. Analogously let for each $j \in J$

$$f_j^{(k)} := \max\left\{t + d_{o,m} | o \in O_j \text{ and } (o, m, t) \in V^{(k+1)}\right\} \tag{36}$$

the finishing time of each job.

6) Using $V^{(k+1)}$, update the next iterates accordingly:
   (a) For all $j \in J$, set

$$s_j^{(k+1)} := \max\left\{c(o) | o \in O_j \text{ and } (o, m, t) \in V^{(k+1)}\right\} + 1, \tag{37}$$

   respectively $s_j^{(k+1)} = 1$ if the set on the right hand side is empty.

   (b) Set $g^{(k+1)} := \min_{m \in M} f_m^{(k)}$.
   (c) Update availability times by

$$a_m^{(k+1)} := f_m^{(k)} - g^{(k+1)} \tag{38}$$

   and

$$a_j^{(k+1)} := \max\left\{0, f_j^{(k)} - g^{(k+1)}\right\}. \tag{39}$$

7) Continue from step 2 with $k \leftarrow k + 1$.

The idea of the above approach is that good solutions for smaller subproblems with less jobs and less operations might not necessarily be part of a good solution for the original problem, as

following operations can drastically change the picture. However, often in practice 'early' operations in a good solution of the smaller subproblem tend to stay rigid when compared to good solutions of the original problem. Hence in each iteration of the above process a tail is cut off from the current solution, thus merely using the start of the solution to the smaller subproblem.

In the following, two strategies are presented how to choose the subset $\widetilde{J}^{(k)} \subseteq R^{(k)}$ of unfinished jobs in the $k$ -th iteration, as well as $u_j^{(k)} \in \left\{s_j^{(k)}, \ldots, n_j\right\}$ for each $j \in \widetilde{J}^{(k)}$.

*Rolling horizon*

With this strategy a *step size* $w \in \mathbb{N}_{>0}$ is defined and all jobs are always handled at the same time. I.e. in the $k$-th iteration define $\widetilde{J}^{(k)} := R^{(k)}$ to be the set of all unfinished jobs of $J$. Then given a job $j \in \widetilde{J}$ set $u_j^{(k)} := \min\{k \cdot w + 1, n_j\}$ in each iteration of the algorithm above. Hence the full solution builds itself up in a rolling fashion, extending its horizon.

*Bottleneck identification*

In larger problems with many jobs, it is not practical to include all jobs in each iteration, so a choice about which jobs to exclude from the subproblem needs to be made each time. The idea in this approach is to assign a bottleneck factor $b_j^{(k)} > 0$ to each unfinished job $j \in \mathbb{R}^{(k)}$ in the $k$ -th iteration. These weight factors should reflect how much the job contributes to the overall makespan of the problem. The greater $b_j^{(k)}$, the more $j$ is considered as a 'bottleneck' and include it in the subproblem, with the aim to start potentially long running jobs early on.

For ease of notation in the following the $(k)$ -superscript in the $k$ -th iteration is omitted.

Given a problem $J$, $O$, $M$ at the start of an iteration in the algorithm, the following steps for each unfinished job $j \in J$ (with usual notation) is performed:

1) Compute the minimum duration that is needed to execute the remaining operations of $j$:

$$d'_{j,\min} := a_j + \sum_{o \in O_j, c(o) \geq s_j} d_{o,\min}. \tag{40}$$

2) Let $q \in \mathbb{N}$ be some predefined number of samples. Each time permuting the jobs randomly, the round-robin algorithm as in chapter 4.7 is performed for $q$ times and obtain $q$ finishing times $f'_{j,1}, \ldots, f'_{j,q}$ of the job $j$. Taking the mean yields an estimated finishing time of $j$ by

$$f'_j := \frac{1}{q} \sum_{i=1}^{q} f'_{j,i}. \tag{41}$$

After $d'_{j,\min}$ and $f'_j$ have been computed for each $j$, a min-max normalization on the values is performed, i.e. for each $j \in R$ set

$$d_{j,\min} := \frac{d'_{j,\min} - \min_{k \in R}\left\{d'_{k,\min}\right\}}{\max_{k \in R}\left\{d'_{k,\min}\right\} - \min_{k \in R}\left\{d'_{k,\min}\right\}} \in [0,1] \tag{42}$$

as well as

$$f_j := \frac{f'_j - \min_{k \in R}\left\{f'_k\right\}}{\max_{k \in R}\left\{f'_k\right\} - \min_{k \in R}\left\{f'_k\right\}} \in [0,1]. \tag{43}$$

In case that $\min_{k \in R}\left\{d'_{k,\min}\right\} = \max_{k \in R}\left\{d'_{k,\min}\right\}$ the normalized minimum duration for job execution $d_{j,\min}$ is set to zero for all $j$; analogously for $f_j$. The bottleneck factor $b_j$ is then calculated as

$$b_j := \sqrt{d_{j,\min}^2 + f_j^2} \in \left[0, \sqrt{2}\right]. \tag{44}$$

The bottleneck factors $b_j$ can be used to choose $\widetilde{J}$ as a predefined number of jobs (e.g. choose $\widetilde{J}$ as the $x$ jobs with the highest bottleneck factor). Furthermore, $b_j$ can also be used to decide how many operations to include for each job $j \in \widetilde{J}$ proportionally to its bottleneck factor (e.g. include more operations for jobs with higher bottleneck factors). In this study, the bottleneck factors are used in the following way.

Given the full problem instance $J$, $O$, $M$, a maximum number $r_J \leq |J|$ of jobs and a desired number $r_O \ll |O|$ of operations to include in each iteration is defined. Furthermore, a minimum number $r_{O,\min}$ of operations to include for each chosen job is selected, with $0 \leq r_{O,\min} \cdot r_J \leq r_O$. The choices in step 2 of the algorithm in chapter 4.8 are made in the following way:

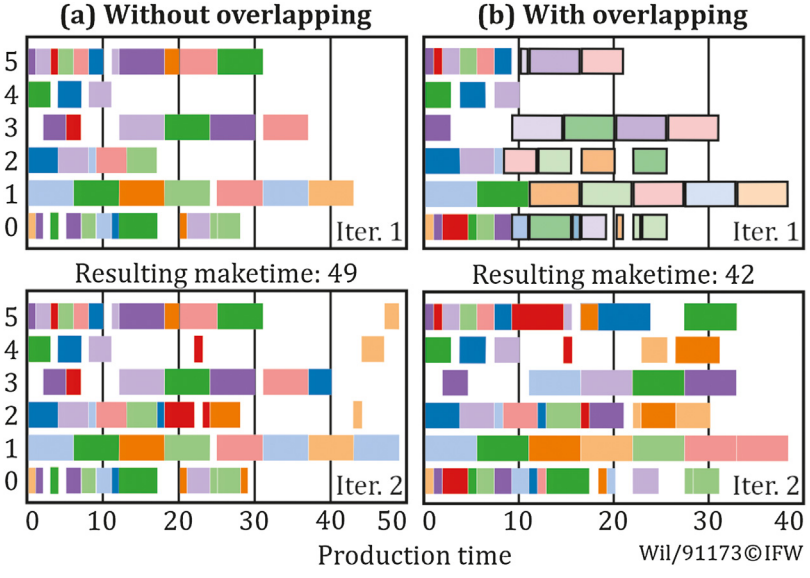


**Fig. 4.** Solving large scale problems without (a) and with (b) overlapped iterations.

- Let $z = |R|$ be the number of unfinished jobs in $J$. Compute the remaining bottleneck factors of all $z$ unfinished jobs as described above. Ordering the jobs in descending order by their bottleneck factors, $\widetilde{J}$ is chosen as the first $\min\{r_J, z\}$ unfinished jobs.
- Let $p := r_O - \left|\widetilde{J}\right| \cdot r_{O,\min}$. For each $j \in \widetilde{J}$ with bottleneck factor $b_j$ and starting operation $s_j$ define

$$\widetilde{u}_j := s_j - 1 + r_{O,\min} + \text{round}\left(\frac{b_j \cdot p}{\sum_{j \in \widetilde{J}} b_j}\right) \tag{45}$$

if $\sum_{j \in \widetilde{J}} b_j \neq 0$, and

$$\widetilde{u}_j := s_j - 1 + r_{O,\min} + \text{round}\left(\frac{p}{\left|\widetilde{J}\right|}\right) \tag{46}$$

otherwise. Then choose $u_j := \min\{\widetilde{u}_j, n_j\}$, and exclude the job $j$ from $\widetilde{J}$ if $u_j < s_j$. Otherwise the operation slice to include in the iteration for job $j$ is defined as

$$\widetilde{O}_j = \{o \in O_j | s_j \leq c(o) \leq u_j\}, \tag{47}$$

as desired.

## Evaluation

The evaluation is split up into three parts. First, the digital annealing model for process parallel flexible job shop scheduling is validated. Therefore, the suitability of the iterative approach for problem decomposition is analyzed. Afterwards, a scientific benchmark using the MK-problems presented in Table 1 evaluates the competitiveness of the new digital annealing model. Further, validation proves the robustness of the digital annealing model for process parallel flexible job shop scheduling against stochastic influences and hyper-parameter variation. Second, the approach will be applied to a practical use-case scenario. A comparison with a MES planning result for this application aims to underline the advantages over currently used PPC methods. Third, a managerial insight is given. Focus here is to demonstrate how the optimization model can easily be connected to other IT-systems via standardized interfaces, which enables a process parallel usage in practice.

### *Validation and scientific benchmark*

For proofing the suitability of the iterative approach for problem decomposition, the mechanism of overlapping iterations (cf. Fig. 3, step four) is of special interest. The Gantt diagrams in Fig. 4 show two different planning results for the problem instance MK-01 with 10 jobs on six machines (0–5) and a total of 55 operations.

In both cases the bottleneck identification method for problem decomposition is applied and the problem has been solved in two iterations, choosing only 44 operations in the first iteration. In the first diagram no operations were removed after the first iteration, leading to a solution with a makespan of 49. In the second diagram a large portion of trailing operations (marked with a black border) is removed, leading to a solution with a makespan of 42, which means that the makespan can be reduced as intended by overlapping iterations.

For the scientific benchmark the hyper-parameters shown in Table 2 are used. These have been iteratively determined as the most suitable parameters during intensive test runs. It can be seen that for the problem MK-09 the rolling horizon method is used. All other problem instances are optimized by using bottleneck identification. With a makespan of 67, the bottleneck identification also provides the best result for MK-09, but the standard deviation of 1.30 for five repeated optimization runs is slightly higher than with rolling horizon (cf. Table 3).

**Table 2**
Model parameterization for scientific benchmark.

| MK | $\tau$ | $\alpha$ | $\beta$ | $\gamma$ | $\delta$ | $Q$ | $\varepsilon$ | $\mu$ | $w$ | $r_J$ | $r_O$ | $r_{O,\min}$ | $q$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 01 | 900 | 800 | 700 | 700 | 200 | 8.192 | 1.0 | 0.4 | – | 10 | 30 | 1 | 300 |
| 02 | 900 | 800 | 700 | 700 | 200 | 8.192 | 1.0 | 0.5 | – | 10 | 30 | 1 | 300 |
| 03 | 900 | 800 | 700 | 700 | 200 | 8.192 | 1.3 | 0.2 | – | 15 | 60 | 1 | 300 |
| 04 | 900 | 800 | 700 | 700 | 200 | 8.192 | 1.0 | 0.6 | – | 15 | 60 | 1 | 300 |
| 05 | 900 | 800 | 700 | 700 | 200 | 8.192 | 1.0 | 0.4 | – | 15 | 50 | 1 | 300 |
| 06 | 900 | 800 | 700 | 700 | 200 | 8.192 | 1.0 | 0.4 | – | 10 | 70 | 5 | 300 |
| 07 | 900 | 800 | 700 | 700 | 200 | 8.192 | 1.0 | 0.4 | – | 20 | 60 | 1 | 300 |
| 08 | 900 | 800 | 700 | 700 | 200 | 8.192 | 1.2 | 0.3 | – | 20 | 50 | 1 | 300 |
| 09 | 900 | 800 | 700 | 700 | 200 | 8.192 | 1.1 | 0.3 | 2 | – | – | – | – |
| 10 | 900 | 800 | 700 | 700 | 200 | 8.192 | 1.2 | 0.5 | – | 20 | 50 | 1 | 300 |

Generally, the results presented in Table 3 show that for MK-03 and MK-08 the best known solution could be confirmed. In 7 out of 10 scenarios the Digital Annealer was able to achieve a solution quality which is higher or equal to the average value of the solutions from Table 1. This is especially true for the larger and thus more practice-relevant problem instances MK-05 to MK-10. Nine out of ten Digital Annealer solutions dominate the results of Ho et al. [24], none of the solutions found is worse. This is of special interest because the approach of Ho et al. [24] basically represents an alternative for process-parallel use in the context of adaptive PPC, due to the very low computational times.

It is noticeable that the best results for the smallest problem instances MK-01 and MK-02 were also achieved by splitting up the production program (three and four iterations), although optimization would have been possible without splitting up. A possible reason may be that many different combinations of work plans and order sequences lead to similar makespan results. In this case, the correct positioning of a few work steps determines the quality of the solution. By dividing the search space into smaller units, the width of the search space is better searched and the solution quality can be improved. In many cases, however, this procedure prevents the absolute optimum from being found. This in turn explains the below-average optimization result for MK-01.

The anneal times are less than 30 s for all problem instances. The calculation time increases linearly with the number of iterations required. In this case, the average calculation time per iteration is 2.69 s with a standard deviation of only 0.05 s. Since obviously the number of required iterations increases significantly slower than the size of the problem, it can be assumed that even for larger optimization problems it is possible to realize process parallel executable computing times. For example, the computing time for problem instance MK-10 is only three times higher than the anneal time for problem MK-01. This is a decisive advantage over the approaches known from literature, where the computing time increases four to 5329 times, according to an overview in Li and Gao [19], enriched by the results from Wang et al. [8], Li et al. [30], Zhang et al. [10] and Nouri et al. [33]. On average of all references mentioned, the computing time increases by a factor of 467.

In order to make first statements about the practical suitability of the new approach, the robustness of the solution quality is examined more closely in the following. In order to be able to make statements about stochastic influences, the optimization runs are first repeated five times with the hyper-parameters from Table 2. The standard deviation of 0.00–3.91 indicates very stable optimization results (cf. Table 3).

In addition, the robustness with regard to hyper-parameter changes is examined as the best settings are normally not known upfront. For this purpose, the ratio of operations to remove from a

**Table 3**
Results of scientific benchmark.

| | Benchmark solution[a] | Annealer solution | Standard deviation | Anneal time | No. of iterations |
|---|---|---|---|---|---|
| MK-01 | 37 / 40.16 | 41 | 0.45 | 7.80 | 3 |
| MK-02 | 26 / 27.11 | 27 | 0.55 | 10.34 | 4 |
| MK-03 | 204 / 204 | 204 | 0.00 | 10.90 | 4 |
| MK-04 | 60 / 63.37 | 67 | 1.10 | 8.15 | 3 |
| MK-05 | 172 / 174.5 | 176 | 0.71 | 8.09 | 3 |
| MK-06 | 57 / 66.11 | 62 | 1.30 | 10.73 | 4 |
| MK-07 | 139 / 144.20 | 144 | 3.27 | 8.10 | 3 |
| MK-08 | 523 / 523.2 | 523 | 0.00 | 21.53 | 8 |
| MK-09 | 307 / 316.3 | 314 | 1.3 | 18.96 | 7 |
| MK-10 | 197 / 222.9 | 214 | 3.91 | 27.06 | 10 |

[a] Best known upper bound / medium value of solutions presented in Table 1.

previously optimized subproblem $\mu$, the maximum number of jobs per subproblem $r_J$, the starting temperature of the annealing process $\tau$ and the weighting factor of the makespan penalty $\delta$ are independently varied against the previously determined optimum value. Only the MK-10 problem instance will be considered, as this is the most practice-oriented one due to the high number of machines and orders. In addition, MK-10 shows a high standard deviation with respect to the mean makespan compared to the other problem instances (cf. Table 3). It is therefore assumed that in this case the optimization model reacts particularly sensitively to changed hyper-parameters.

Fig. 5 shows that the increase in makespan is usually small (<10%) or very small (<5%) when the selected hyper-parameters are varied. The lower limit of the 95% confidence interval is even slightly below zero in five of eight cases and with 1.68% and 0.87% only a little in the positive range for $r_J = 15$ respectively $\mu = 0.2$. Due to the low standard deviation in repeated optimization runs, the upper limits of the confidence intervals are also below a makespan deviation of 10% in six out of 10 cases. For $r_J = 15$, a probability of error of 5% results in a moderately higher maximum makespan deviation of 12.15%. Only for $r_J = 10$, which corresponds to a deviation of 50% from the previously selected ideal setting, a strong increase in makespan of 17.5% on average can be registered.

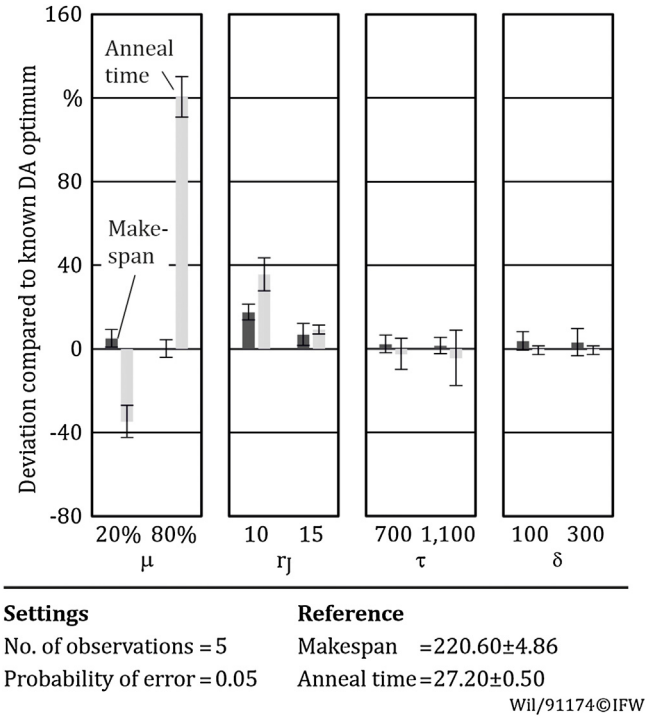


**Fig. 5.** Impact of hyper-parameter variations on realized makespan and anneal time.

Overall, the results indicate that the optimization model is robust to hyper-parameters variations with regard to the makespan. However, the average 3.72% deterioration of the optimization result at $\delta = 300$ indicates that the makespan objective $H_3$ (cf. Eq. 22) is not ideally suited for the present problem. Due to the higher weighting of $H_3$, invalid solutions would have to occur more frequently with previously ideal model parameterization. Alternatively, better results should be obtained if the choice of $\delta = 200$ was not optimal. The measured deterioration suggests that the current penalization of each delay of a single operation does not fully correlate with the actual goal of makespan reduction. In fact, if the values for $\delta$ are too high, individual operations are shifted in positions that are unfavorable for the makespan, which means that the local optimization slows down the global one.

As already shown before, the anneal time correlates with the number of required iterations. This in turn is on average almost independent of the hyper-parameters $\tau$ and $\delta$, which is why no significant effect on the anneal time can be demonstrated for a variation of both parameters. In contrast, it becomes clear that the correct choice of $\mu$ and $r_J$ influence the anneal time significantly. Is $\mu$ set too high the anneal time increases greatly (120.33% in the above example), while the makespan is not reduced significantly.

*Practical application*

For a more practical evaluation, the digital annealing approach is investigated on the basis of the production of a sample component. The production program foresees the production of 40 jobs of one product type with different lot sizes (ranging from 200 to 680 pieces). In total six machines are available for production, which differ in terms of the work steps they can execute and the processing speed for each workpiece (cf. Fig. 6). A standardized format was developed to map these characteristics. In this format, the relevant technological relationships can be depicted using product-specific machine matrices, and lot size-independent basic processing times can be stored. Empty cells indicate that a machine is not suitable for processing a machine-work step combination of the product type in question. Order quantities (lot sizes) and a product type identifier (pID) are added to each job which is to be scheduled. The pIDs are used as link to the product-specific machine matrices (see chapter 5.3 for more details).

The evaluation of the solution quality is done by comparison with two different planning results from a MES. The first one uses a simple forward scheduling algorithm without holistic optimization. A work step of an order to be planned is always allocated to the next free machine that is technically suitable for this work step. The sequence of the operation allocation is based on the delivery dates of the orders. Since these were all assumed to be the same in

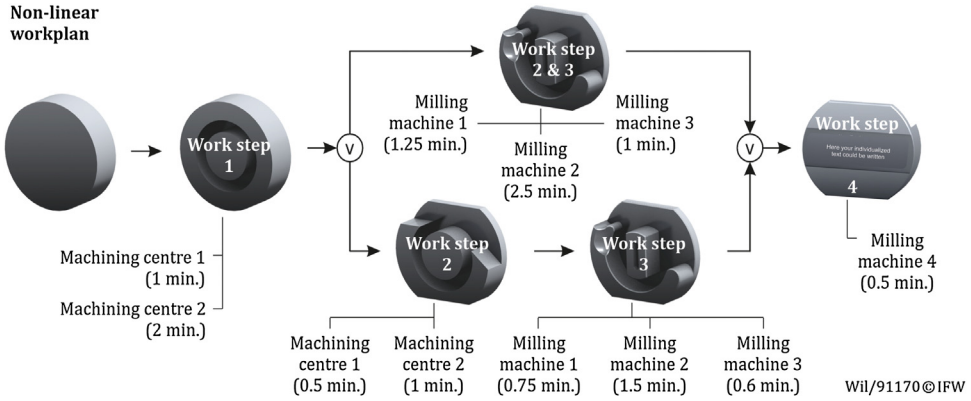


**Fig. 6.** Non-linear workplan of sample component production.

the present example, the selection is made in ascending order according to the order numbers. The low computing time of this simple scheduling algorithm makes process-parallel execution easy.

In the second reference scenario, the MES is extended via a Plug-In to include the Genetic Algorithm of Denkena et al. [2] for holistically optimized machine allocation (cf. chapter 2). Sequencing in front of the machines is still performed by the rule-based planning algorithm of the MES. Within this approach the computing time depends on the selected number of individuals and generations. A process-parallel execution can thus be limited.

In Table 4 the makespan result of all three approaches is benchmarked. The corresponding hyper-parameter settings for the digital annealing approach are shown in Table 5 (exp. V0). Compared to pure MES planning, the makespan can be reduced by 138.2 h with the new approach. This corresponds to a saving of 40.90% in relation to the procedure currently used in practice. The use of the Genetic Algorithm for holistically optimized work plan selection, on the other hand, only achieves a makespan reduction of 18.32% (or 61.9 h) compared to the conventional MES solution. Consequently, for practical application it can be concluded that the new approach is superior to the existing approaches both in terms of computing time and makespan optimization.

Also within the scope of the practical evaluation, a sensitivity analysis was carried out. In order to investigate to what extent the findings on the robustness of the optimization approach gained in the preceding chapter can be confirmed, the hyper-parameter $\mu$ and $r_J$ were again varied compared to the settings identified as ideal. For all the three experiments V0-V2, the same configurations as for the MK-problems were used for the starting temperature, the weighting factors of the constraints $H_0$, $H_1$ and $H_2$ as well as for the makespan objective $H_3$ ($\tau = 900$, $\alpha = 800$, $\beta = \gamma = 700$, $\delta = 200$). The maximum number of possible binary variables $Q$ is also fixed at 8192 again. The other hyper-parameters used are presented in Table 5. Note that $r_{o,\min}$ is set to the maximum of four operations, which means that with $r_O = 32$, respectively $r_O = 40$ it is ensured that the maximum of eight, respectively 10 jobs per iteration are considered for all experiments (V0-V2).

**Table 4**
Makespan evaluation for different optimization approaches.

| | DA | GA + MES | MES |
|---|---|---|---|
| Min. makespan | 199.68h | 275.98h | 337.88h |
| DA | – | −27.65% | −40.90% |
| GA + MES | +38.21% | – | −18.32% |
| MES | +69.21% | +22.43% | – |

DA: Digital Annealing Approach; GA + MES: MES with Genetic Algorithm Plug-In.

**Table 5**
Model parametrization for sensitivity analysis.

| Exp. | $\varepsilon$ | $\mu$ | $r_J$ | $r_O$ | $r_{O,\min}$ | $q$ |
|---|---|---|---|---|---|---|
| V0 | 1.2 | 0 | 8 | 32 | 4 | 300 |
| V1 | 1.2 | 0.3 | 8 | 32 | 4 | 300 |
| V2 | 1.2 | 0 | 10 | 40 | 4 | 300 |

The results presented in Table 6 show that the robustness of the new optimization approach against hyper-parameter deviation can be confirmed with regard to the maximum number of jobs to be used per iteration in the practical application scenario (experiment V2). An increase of $r_J$ by 25% compared to experiment V0 results only in an increase of the mean makespan of 5.59%, which is still far below the results the MES-based approaches (cf. Table 4). The minimum makespan at experiment V2 is even only 1.95% above that of the benchmark experiment V0. But, in comparison to the scientific benchmark the new optimization model reacts more sensitively to a significant increase in the ratio of operations to remove to $\mu = 0.3$. Here the average makespan increases by 17.06% compared to the V0 settings, the minimum makespan is 17.74% higher. This finding stands in contrast to the results of the scientific analysis chapter 5.1, where choosing $\mu$ too high did not significantly change the makespan (cf. Fig. 5). A possible reason for this model behavior is the high number of jobs with a low number of operations per job. This constellation means that entire jobs are already removed from the existing schedule for $\mu = 0.3$ and must therefore be completely rescheduled in the next iteration. As a result, the previously performed optimization effort for this entire job is lost and at the same time it becomes more difficult to optimally integrate the first operations of the job into the already existing schedule again. Nevertheless, the results basically support the findings from the scientific benchmark, where the strongly increased anneal time for $\mu = 0.8$ also indicated that choosing too high values for $\mu$ is not desirable. As a result, a conservative approach should always be followed when choosing $\mu$ and lower settings should be preferred initially. Values of $\mu > 0.5$ do not seem to be appropriate in principle.

However, the results generally also indicate that at least for the hyper-parameters of the sub-method for iteratively solving large-scale problems $\mu$, $r_J$, $r_O$ and $r_{O,\min}$ a problem-specific adaptation must always be made. As a result, no clear guidelines for the parameterization of the digital annealing approach can be given, at least according to the current state of knowledge. Nevertheless, the relatively small number of particularly relevant parameters and the fact that obviously a certain parameterization window always produces good optimization results allow a practical application.

**Table 6**
Results of sensitivity analysis.

| Exp. | Minimum makespan | Avg. makespan[a] | Standard deviation[a] | Avg. anneal time[a] | Avg. no. of iterations[a] |
|---|---|---|---|---|---|
| V0 | 199.68h | 203.26h | 4.27 | 13.59 | 5.00 |
| V1 | 235.10h | 237.94 | 3.00 | 25.26 | 9.40 |
| V2 | 203.58h | 214.62 h | 6.82 | 12.97 | 4.80 |

[a] Out of five repeated optimization runs.

*Managerial insights*

The approach presented above can only be transferred into practice if a simple integration into the existing PPC-IT-infrastructure is possible. The concept shown in Fig. 7, which has been aligned with a MES manufacturer, shows that this is possible for the given approach. The needed job and product type identifier (jID, pID), the lot size and a work step identifier (wsID) as well as the possible machines (mID) for each work step can be exported out of the MES to a. txt-file via SQL query. A method for the automated identification of alternative machine allocation options, based on job and machine characteristics as well as an ontology based approach is described in Denkena et al. [2].

The exported data can be automatically transferred into the standardized input format described in chapter 5.2 (e.g. using a Visual Basic for Application (VBA) script in combination with the common spreadsheet software EXCEL). The transfer of the optimization problem to the Digital Annealer as well as the return of the optimization results is done with standard commands from open source Python libraries. The selected. txt output format in turn allows the planning results to be fed back directly into the SQL database of the MES. The program-independent transfer format again allows easy integration into alternative or further software systems (e.g. EXCEL or databases of other MES manufacturers).

## Conclusion and outlook

Modern production planning and control strategies require continuous and process-parallel optimization of machine allocation and processing sequences. Therefore, this paper presented a new method for process parallel flexible job shop scheduling using the concept of quantum annealing based optimization. The results of a scientific benchmark showed that the new approach provides results similar to or equal to those of the classical heuristics, but at the same time delivers reliably low anneal times, especially for large problem instances. Thus, for a FJSP with 20 jobs and 15 machines in 27.06 s, a solution could be found that undercuts the average makespan of 19 classical reference models by about 4%. Further evaluations showed that the solutions of the new approach are robust against stochastic influences. Even hyper-parameter variations have little effect on the achieved makespan optimization, and if they do, then only when there are high deviations from the ideal setting. The application within the scope of a practical use case confirms the good performance of the new approach. Compared to a classical MES solution and a combination of MES and genetic algorithm, significant makespan reductions of 28% and 41% respectively were achieved.

Finally, the investigations within this paper showed that the makespan objective formulation $H_3$ is not ideally chosen for makespan optimization. In first additional experiments the best results of the round-robin algorithm were used for the formulation of an additional binary objective polynomial $H_4$ which quadratically penalizes late operation finishing times on each machine individually. Thus, a minimum makespan of 196 h was achieved for the sample component production from Fig. 6. For MK-10, a makespan of 209 was obtained with the alternative approach. In three repeated observations, the standard deviation was reduced to 1.73 and thus more than halved compared to the results from Table 3. Also for MK-09 a better makespan of 311 was achieved, but both the standard deviation (2.08) and the number of required

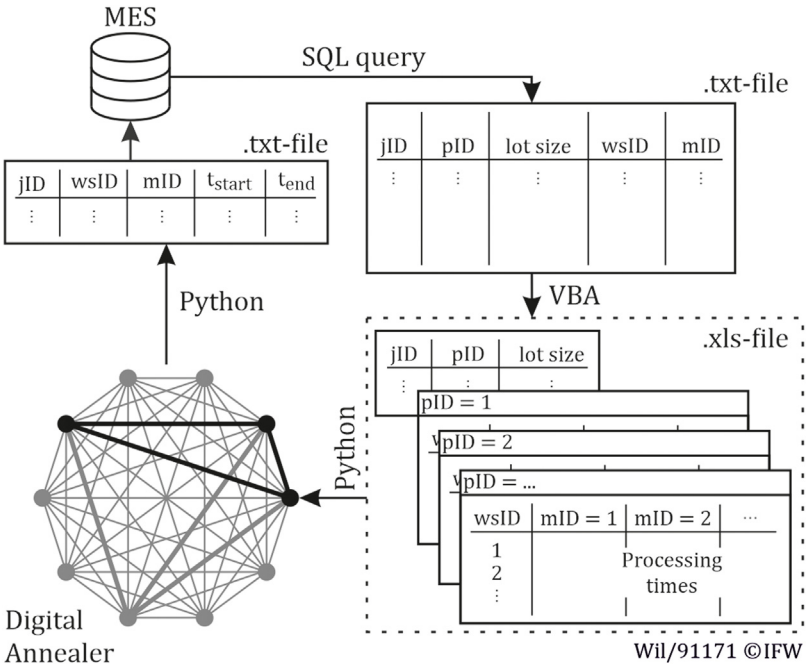


**Fig. 7.** Concept for MES integration.

iterations (10) and thus the anneal time (27.07 s) increased. It is necessary to carry out additional research and to further improve the objective polynomial formulation if possible.

In further investigations, additional aspects for practical application must be integrated into the QUBO formulation. This includes, for example, the consideration of non-productive times (e.g. setup) and shift calendars as well as the extension of the objective function by additional optimization objectives. Here, in a first step, the capacity utilization is a further target criterion, since previous investigations of the MK problems show that the same makespan can be realized at different workload conditions (e.g. [28]). Additionally, a focus will be put on the optimization of the QUBO build time, which has not been considered so far.

## Declaration of interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

The authors acknowledge the financial support by the Federal Ministry for Economic Affairs and Energy of Germany in the project IIP-Ecosphere (project number 01MK20006A).

## References

[1] Denkena, B., Lorenzen, L.-E., Schmidt, J., 2012, Adaptive Process Planning. Production Engineering, 6/1: 55–67. http://dx.doi.org/10.1007/s11740-011-0353-7.
[2] Denkena, B., Wilmsmeier, S., Stock, W., 2017, Betriebsbegleitende Adaptive Arbeitsplanung und Fertigungssteuerung. VDI-Z, 159/10: 30–32.
[3] Denkena, B., Winter, F., Wilmsmeier, S., 2019, Machine Learning in der Adaptiven Fertigungssteuerung - Genetischer Algorithmus zur Bewertung Alternativer Arbeitspläne. Fabriksoftware, 24/3: 17–20.
[4] Brandimarte, P., 1993, Routing and Scheduling in a Flexible Job Shop by Tabu Search. Annals of Operations Research, 41:157–183. http://dx.doi.org/10.1007/BF02023073.
[5] Garey, M.R., Jhonson, D.S., Sethi, R., 1976, The Complexity of Flowshop and Jobshop Scheduling. Mathematical Methods of Operations Research (Heidelberg Germany), 1/2: 117–129. http://dx.doi.org/10.1287/moor.1.2.117.
[6] Pezzella, F., Morganti, G., Ciaschetti, G., 2008, A Genetic Algorithm for the Flexible Job-Shop Scheduling Problem. Computers & Operations Research, 35/10: 3202–3212. http://dx.doi.org/10.1016/j.cor.2007.02.014.
[7] Bagheri, A., Zandieh, M., Mahdavi, I., Yazdani, M., 2010, An Artificial Immune Algorithm for the Flexible Job-Shop Scheduling Problem. Future Generations Computer Systems: FGCS, 26/4: 533–541. http://dx.doi.org/10.1016/j.future.2009.10.004.
[8] Wang, X., Gao, L., Zhang, C., Shao, X., 2010, A Multi-Objective Genetic Algorithm Based on Immune and Entropy Principle for Flexible Job-Shop Scheduling Problem. Int J Adv Manuf Technol, 51:757–767. http://dx.doi.org/10.1007/s00170-010-2642-2.
[9] De Giovanni, L., Pezzella, F., 2010, An Improved Genetic Algorithm for the Distributed and Flexible Job-Shop Scheduling Problem. European Journal of Operational Research, 200/2: 395–408. http://dx.doi.org/10.1016/j.ejor.2009.01.008.
[10] Zhang, G., Gao, L., Shi, Y., 2011, An Effective Genetic Algorithm for the Flexible Job-Shop Scheduling Problem. Expert Systems with Applications, 38/4: 3563–3573. http://dx.doi.org/10.1016/j.eswa.2010.08.145.
[11] Teekeng, W., Thammano, A., 2012, Modified Genetic Algorithm for Flexible Job-Shop Scheduling Problems. Procedia Computer Science, 12:122–128. http://dx.doi.org/10.1016/j.procs.2012.09.041.
[12] Alcan, P., Basligi, H., 2012, A Genetic Algorithm Application using Fuzzy Processing Times in Non-Identical Parallel Machine Scheduling Problem. Advances in Engineering Software (Barking London England: 1992), 45/1: 272–280. http://dx.doi.org/10.1016/j.advengsoft.2011.10.004.
[13] Chiang, T.C., Lin, H.J., 2013, A Simple and Effective Evolutionary Algorithm for Multiobjective Flexible Job Shop Scheduling. International Journal of Production Economics, 141/1: 87–98. http://dx.doi.org/10.1016/j.ijpe.2012.03.034.
[14] Driss, I., Mouss, K.N., Laggoun, A., 2015, An Effective Genetic Algorithm for the Flexible Job Shop Scheduling Problems. 11th Congres International de Genie Industriel-GIGI2015 Québec Canada, .
[15] Sun, L., Lin, L., Wang, Y., Gen, M., Kawakami, H., 2015, A Bayesian Optimization-Based Evolutionary Algorithm for Flexible Job Shop Scheduling. Procedia Computer Science, 61:521–526. http://dx.doi.org/10.1016/j.procs.2015.09.207.
[16] Huang, X., Guan, Z., Yang, L., 2018, An Effective Hybrid Algorithm for Multi-Objective Flexible Job-Shop Scheduling Problem. Adv Mech Eng, 10/9: 1–14. http://dx.doi.org/10.1177/1687814018801442.
[17] Dalfard, V.M., Mohammadi, G., 2012, Two Meta-Heuristic Algorithms for Solving Multi-Objective Flexible Job-Shop Scheduling with Parallel Machine and Maintenance Constraints. Computers & Mathematics with Applications (Oxford England: 1987), 64/6: 2111–2117. http://dx.doi.org/10.1016/j.camwa.2012.04.007.
[18] Rossi, A., Boschi, E., 2009, A Hybrid Heuristic to Solve the Parallel Machines Job-Shop Scheduling Problem. Advances in Engineering Software (Barking London England: 1992), 40/2: 118–127. http://dx.doi.org/10.1016/j.advengsoft.2008.03.020.
[19] Li, X., Gao, L., 2016, An Effective Hybrid Genetic Algorithm and Tabu Search for Flexible Job Shop Scheduling Problem. International Journal of Production Economics, 174:93–110. http://dx.doi.org/10.1016/j.ijpe.2016.01.016.
[20] Zhang, H., Manier, H., Manier, A., 2012, A Genetic Algorithm with Tabu Search Procedure for Flexible Job Shop Scheduling with Transportation Constraints and Bounded Processing Times. Computers & Operations Research, 39/7: 1713–1723. http://dx.doi.org/10.1016/j.cor.2011.10.007.
[21] Kacem, I., Hammadi, S., Borne, P., 2002, 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: A Publication of the IEEE Systems Man and Cybernetics Society, 32/1: 1–13. http://dx.doi.org/10.1109/TSMCC.2002.1009117.
[22] Kacem, I., Hammadi, S., Borne, P., 2002, Pareto-Optimality Approach for Flexible Job-Shop Scheduling Problems: Hybridization of Evolutionary Algorithms and Fuzzy Logic. Mathematics and Computers in Simulation, 60/3-5: 245–276. http://dx.doi.org/10.1016/S0378-4754(02)00019-8.
[23] Ho, N.B., Tay, J.C., 2004, GENACE: An Efficient Cultural Algorithm for Solving the Flexible Job-Shop Problem. in: Proceedings of the 2004 Congress on Evolutionary Computation (Portland, USA), pp.1759–1766. http://dx.doi.org/10.1109/CEC.2004.1331108.
[24] Ho, N.B., Tay, J.C., Lai, E.M.-K., 2007, An Effective Architecture for Learning and Evolving Flexible Job-Shop Schedules. European Journal of Operational Research, 179/2: 316–333. http://dx.doi.org/10.1016/j.ejor.2006.04.007.
[25] Li, J.-Q., Pan, Q.-K., Liang, Y.-C., 2010, An Effective Hybrid Tabu Search Algorithm for Multi-Objective Flexible Job-Shop Scheduling Problems. Comput Ind Eng, 59/4: 647–662. http://dx.doi.org/10.1016/j.cie.2010.07.014.
[26] Xing, L.-N., Chen, Y.W., Wang, P., Zhao, Q.S., Xiong, J., 2010, A Knowledge-Based ant Colony Optimization for Flexible Job Shop Scheduling Problems. Applied Soft Computing, 10/3: 888–896. http://dx.doi.org/10.1016/j.asoc.2009.10.006.
[27] Chiang, T.C., Lin, H.J., 2012, Flexible Job Shop Scheduling Using a Multiobjective Memetic Algorithm. in Huang DD.S.S, Gan YY., Gupta PP., Gromiha MM.M.M, (Eds.) Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence. ICIC 2011. Lecture Notes in Computer Science, vol 6839. Springer, Berlin, Heidelberg. pp. pp.49–55. http://dx.doi.org/10.1007/978-3-642-25944-9_7.
[28] Moslehi, G., Mahnam, M., 2011, A Pareto Approach to Multi-Objective Flexible Job-Shop Scheduling Problem Using Particle Swarm Optimization and Local Search. International Journal of Production Economics, 129/1: 14–22. http://dx.doi.org/10.1016/j.ijpe.2010.08.004.
[29] Huang, S., Tian, N., Wang, Y., Ji, Z., 2016, Multi-Objective Flexible Job-Shop Scheduling Problem using Modified Discrete Particle Swarm Optimization. SpringerPlus, 5:1432. http://dx.doi.org/10.1186/s40064-016-3054-z.
[30] Li, J.-Q., Pan, Q.-K., Suganthan, P.N., Chua, T.J., 2011, A Hybrid Tabu Search Algorithm with an Efficient Neighborhood Structure for the Flexible Job Shop Scheduling Problem. The International Journal of Advanced Manufacturing Technology, 52:683–697. http://dx.doi.org/10.1007/s00170-010-2743-y.
[31] Shen, L., Dauzère-Pérès, S., Neufeld, J.S., 2018, Solving the Flexi-Ble Job Shop Scheduling Problem with Sequence-Dependent Setup Times. European Journal of Operational Research, 256/2: 503–516. http://dx.doi.org/10.1016/j.ejor.2017.08.021.
[32] Xing, L.-N., Chen, Y., Yang, K., 2009, An Efficient Search Method for Multi-Objective Flexible Job Shop Scheduling Problems. Journal of Intelligent Manufacturing, 20:283–293. http://dx.doi.org/10.1007/s10845-008-0216-z.
[33] Nouri, H.E., Driss, O.B., Ghédira, K., 2015, Hybrid Metaheuristics within a Holonic Multiagent Model for the Flexible Job Shop Problem. Procedia Computer Science, 60:83–92. http://dx.doi.org/10.1016/j.procs.2015.08.107.
[34] Colwell, R., 2013, The Chip Design Game at the End of Moore'S Law. 2013 IEEE Hot Chips 25 Symposium (HCS), Stanford, CA: 1–16. http://dx.doi.org/10.1109/HOTCHIPS.2013.7478302.
[35] Lucas, A., 2014, Ising Formulations of Many NP Problems. Frontiers in Physics, 2/5: 1–27. http://dx.doi.org/10.3389/fphy.2014.00005.
[36] Kadowaki, T., Nishimori, H., 1998, Quantum Annealing in the Transverse Ising Model. Physical Review E, 58/5: 5355–5363. http://dx.doi.org/10.1103/PhysRevE.58.5355.
[37] D-Wave. 2011, D-Wave Systems sells its first Quantum Computing System to Lockheed Martin Corporation. (Accessed 07 May 2020)*https://www.dwavesys.com/news/d-wave-systems-sells-its-first-quantum-computing-system-lockheed-martin-corporation*.
[38] Tsukamoto, S., Takatsu, M., Matsubara, S., Tamura, H., 2017, An Accelerator Architecture for Combinatorial Optimization Problems. Fujitsu Sci Tech J, 53/5: 8–13.
[39] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E., 1953, Equation of State Calculations by Fast Computing Machines. The Journal of Chemical Physics, 21/6: 1087–1092. http://dx.doi.org/10.1063/1.1699114.

[40] Kirkpatrick, S., Gelatt Jr C.D., Vecchi, M.P., 1983, Optimization by Simulated Annealing. Sci, 220/4598: 671–680. http://dx.doi.org/10.1126/science.220.4598.671.
[41] Matsubara, S., Takatsu, M., Miyazawa, T., Shibasaki, T., Watanabe, Y., Takemoto, K., Tamura, H., 2020, Digital Annealer for High-Speed Solving of Combinatorial Optimization Problems and Its Applications. 2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC) (Beijing, China), pp.667–672. http://dx.doi.org/10.1109/ASP-DAC47756.2020.9045100.
[42] Perdomo-Ortiz, A., Feldmann, A., Ozaeta, A., Isakov, S.V., Zhu, Z., O'Gorman, B., Katzgraber, H.G., Diedrich, A., Neven, H., de Kleer, J., Lackey, B., Biswas, R., 2019, Readiness of Quantum Optimization Machines for Industrial Applications. Phys Rev Applied, 12/1: 014004. http://dx.doi.org/10.1103/PhysRevApplied.12.014004.
[43] Venturelli, D., Marchand, D.J.J., Rojo, G., 2016, Quantum Anneling Implementation of Job-Shop Scheduling. Quantum Physics Letters, . Cornell University. arXiv:1506.08479v2.
[44] Hurink, J., Jurisch, B., Thole, M., 1994, Tabu Search for the Job-Shop Scheduling Problem with Multi-Purpose Machines. OR Spektrum, 15:205–215. http://dx.doi.org/10.1007/BF01719451.
[45] Khoukhi, F.E., Boukachour, J., Hilali Alaoui, A.E., 2017, The "Dual-Ants Colony": A Novel Hybrid Approach for the Flexible Job Shop Scheduling Problem with Preventive Maintenance. Comput Ind Eng, 106:236–255. http://dx.doi.org/10.1016/j.cie.2016.10.019.
[46] Liu, H., Abraham, A., Wang, Z., 2009, A Multi-Swarm Approach to Multi-Objective Flexible Job-Shop Scheduling Problems. Fundamenta Informaticae, 94/4: 465–489. http://dx.doi.org/10.3233/FI-2009-160.
[47] Mastrolilli, M., Gambardella, L.M., 2000, Effective Neighbourhood Functions for the Flexible Job Shop Scheduling Problem. J Sched, 3/1: 3–20. http://dx.doi.org/10.1002/(SICI)1099-1425(200001/02)3:1<3::AID-JOS32>3.0.CO;2-Y.
[48] Nouiri, M., Bekrar, A., Jemai, A., Trentesaux, D., Ammari, A.C., Niar, S., 2017, Two Stage Particle Swarm Optimization to Solve the Flexible Job Shop Predictive Scheduling Problem Considering Possible Machine Breakdowns. Comput Ind Eng, 112:595–606. http://dx.doi.org/10.1016/j.cie.2017.03.006.
[49] Phanden, R.K., Jain, A., Verma, R., 2011, Integration of Process Planning and Scheduling: A State-of-the-Art Review. International Journal of Computer Integrated Manufacturing, 24/6: 517–534. http://dx.doi.org/10.1080/0951192X.2011.562543.
[50] Rossi, A., Dini, G., 2007, Flexible Job-Shop Scheduling with Routing Flexibility and Separable Setup Times using ant Colony Optimi-Sation Method. Robotics and Computer- Integrated Manufaturing, 23/5: 503–516. http://dx.doi.org/10.1016/j.rcim.2006.06.004.
[51] Xia, W., Wu, Z., 2005, An Effective Hybrid Optimization Approach for Multi-Objective Flexible Job-Shop Scheduling Problems. Computers and Industrial Engineering, 48/2: 409–425. http://dx.doi.org/10.1016/j.cie.2005.01.018.
[52] Zhang, G., Shao, X., Li, P., Gao, L., 2009, An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem. Computers Industrial Engineering, 56/4: 1309–1318. http://dx.doi.org/10.1016/j.cie.2008.07.021.