Production Engineering
https://doi.org/10.1007/s11740-022-01152-9

PRODUCTION MANAGEMENT

Hybrid Monte Carlo tree search based multi-objective scheduling

Constantin Hofmann[1] · Xinhong Liu[1] · Marvin May[1] · Gisela Lanza[1]

Received: 20 April 2022 / Accepted: 20 July 2022

Abstract
As markets demand targeted products for highly differentiated use cases, the number of variants in production increases, whilst the volume per variant decreases. Different product variants result in differences in work content on workstation level which cause takt time losses and result in a poor utilization. In this context, matrix-structured production systems with neither temporal nor spacial linkage emerged to reduce the effects of different work content on the entire production system. However, matrix-structured production systems require far more complex production control. To that end, this paper presents a scheduling approach. The proposed scheduling system considers variable process sequences and their allocation to different workstations in order to optimize scheduling objectives. This contribution presents a Monte Carlo tree search based optimizer combined with local search as post optimizer to derive schedules in a short time span to enabling reactive scheduling. The application of the scheduler to a benchmark problem and an industrial scheduling problem demonstrates the quality of the results and illustrates how the scheduler reassigns the work content dynamically.

Keywords Scheduling · Matrix production · Monte Carlo tree search · Production systems

1 Introduction

State-of-the-art production systems strive to reconcile product individualization with short delivery times and the cost and quality of mass production [1]. Product individualization often results in an increased number of variants with different work content per workstation. In this setting, rigid production lines with temporal and spacial linking suffer from a loss of efficiency due to the unequally distributed work content between the different workstations [2]. Subsequently, new concepts are emerging which aim to overcome the takt time dependency with flexible production systems without temporal or spacial linkage. These concepts, known as matrix production [3], line-less assembly [4], modular assembly or cubeTec rely on placing multi-purpose workstations in a matrix-like structure in space without having a common takt time for the entire production system. This flexible production structure opens new possibilities to distribute workload in the production system as the scheduling system can vary the order of the required processes steps within the flexibility that the individual precedence graph of each variant offers [5]. At the same time, the free material flow makes it possible to optimize different production objectives for each order. On the downside, the complexity for the scheduling system increases significantly with each further degree of freedom. To achieve a reactive scheduling that is capable of using these degrees of freedom, highly efficient scheduling approaches are needed. Depending on the market characteristics, different objectives, e.g. minimal delay, high output or high utilization, are relevant for the production. These objectives should be optimized, even in a highly dynamic environment with disturbances in the production system. This paper presents thus a hybrid reactive scheduling approach capable of optimizing different objectives.

In the following, the relevant literature will be presented (Sect. 2) before the hybrid scheduling system is introduced (Sects. 3.1, 3.2) along with means to improve its performance and interaction with the base optimizer (Sect. refsearchstrategy). Section 4 provides the results of the different

✉ Constantin Hofmann
constantin.hofmann@kit.edu

Marvin May
marvin.may@kit.edu

Gisela Lanza
gisela.lanza@kit.edu

[1] wbk Institute of Production Science, Karlsruhe Institute of Technology KIT, Kaiserstraße 12, 76131 Karlsruhe, Baden-Wuerttemberg, Germany

modifications (Sects. 4.2–4.5) and shows the performance of the scheduling system on a benchmark problem as well as on an industrial use case (Sects. 4.6 and 4.7). This paper concludes with a conclusion and outlook.

2 Literature review

The scheduling task of a matrix production system consists of three sub-decisions. First, selecting one of the feasible operations for each product respecting the restrictions defined by the precedence graph. Second, allocating the selected operation to a workstation. Third, scheduling the start of the operation on the selected workstation. These decisions can be taken sequentially or simultaneously. Transport, setup and cycle times as well as the status and possible processes of the workstations are restrictions to the problem. Relevant scheduling objectives are for instance the minimization of tardiness, throughput time or makespan and the maximization of utilization [6].

Scheduling is known as an NP hard problem (non-deterministic polynomial-time) [7]. Therefore, exact optimization approaches are mostly used to generate optimal solutions for static reference problems. Research in this field addresses the challenge of finding efficient problem representations [8]. Since the reactive scheduling task at hand requires a short computing time and a feasible solution at all times, exact optimization procedures are unsuitable and, thus, not discussed further.

Dynamic scheduling addresses the problem of deriving a schedule in the context of dynamic events e.g. when a break-down occurs, a processes takes longer than expected (resource-related event) or a new order arrives (job-related event) [9]. In this context, predictive-reactive scheduling describes a scheduling approach where a schedule is derived based on assumptions of the future. However, when a deviation occurs, the schedule is either partly or completely revised [9]. Due to the resulting time constraints, heuristics play a predominant role.

Problem-specific heuristics, also known as priority rules, are a well-researched field. It could be shown that the choice of the priority rule depends on the optimization objectives [10] and that their performance degrades with increasing system complexity [11]. A matrix production system offers many degrees of freedom, therefore approaches purely based on problem-specific heuristics are not advisable.

Meta-heuristics such as genetic algorithms (GA) or simulated annealing (SA) play an important role for solving highly complex problems. Balancing search speed with the potential pitfall of early convergence against a local minimum and thorough search are very challenging [12]. To counterbalance the global search of a GA, local search is often applied as a post-optimizer to enhance the local search abilities. This combination is known as memetic algorithm [13].

The domain of games also features highly complex markov decision problems and as such has brought forth promising approaches that can be adapted to solve complex problems in other domains such as scheduling. Inspired by the game-play of two opposing players, bi-objective scheduling problems can be modeled as a game between two players competing to optimize their respective objective. This approach has favorable properties regarding complexity growth and thus reduces solution time [14]. Alternatively, the scheduling problem can be modelled as $N + 1$ game where n production orders try to optimize their cycle time and the production system as single player thrives for high utilization [15]. Lately, Monte Carlo tree search (MCTS) has gained a lot of interest as means to solve complex problems in gaming [16] but also in the domain of scheduling [17–19]. Monte Carlo tree search as an algorithm is an iterative anytime approach using search trees to represent Markov decision problems. The four-phase approach consists of a selection phase, in which a node (Markov state) is selected for expansion. In the expansion phase, one of the possible actions is executed. In the following rollout phase, random actions are applied until a terminal state is reached. In the fourth phase, the backpropagation, the reached terminal state is evaluated with respect to the objectives and the nodes on the path to the terminal state are updated accordingly. These four steps are repeated until a time or iteration-based termination criterion is reached [17]. MCTS can be used to solve multi-objective scheduling problems and, as domain-independent approach, does not require the transformation of the problem to a specific form other then a markov decision problem.

Local search designates optimization approaches that evaluate neighboring solutions around an initial solution in the search for an improved result [20]. The neighborhood is defined by a neighborhood structure. Even though there are many different neighborhood structures, in the context of scheduling, most structures rely on shifting and swapping of operations. For scheduling problems, local search is mostly applied to the critical path. The critical path denotes a sequence of operations without a temporal buffer between operations. Any deviation on the critical path leads to an overall deviation [21].

Especially hybrid MO-MCTS approaches have demonstrated their abilities by solving the Pareto Kacem benchmark problem [19, 22, 23]. However, the Kacem benchmark problem does not cover important restrictions (transport, setup) and does not offer process flexibility. Therefore, the MCTS approaches found in literature as well as the local search methods do not cover these restrictions. This paper proposes a hybrid multi-objective MCTS (MO-MCTS) combined with local search (MO-MCTS + LS) tailored for scheduling of a matrix production with transport, setup and process flexibility. The focus lies on the interaction between base-optimizer and local search as well as on the search strategy applied.

3 Approach

The scheduling system consists of two main components. First, the MO-MCTS optimizer that generates a complete schedule and secondly the local search to explore the neighborhood around one or several solutions generated by the MO-MCTS optimizer. In the following, the base optimizer is introduced shortly before the local search is presented as well as several measures to improve the search performance.

3.1 MO-MCTS base optimizer

To model the scheduling problem as search tree, it is transferred into nodes and edges. Each node represents a state which is defined by a schedule with operations allocated to workstations in a specific time periods. Each edge of the graph represents an action of adding one operation to a workstation's schedule.

The optimization run begins with the currently fixed schedule as the starting point. At the beginning of the period the schedule is empty. For subsequent runs, started operations as well as operations that cannot be changed any more are already included in the schedule. MO-MCTS is given an iteration budget until the run terminates. The optimization objective is the maximization of the Pareto front. The Pareto front is quantified by the hypervolume of dominated space under the Pareto points [24]. During selection, the node with the highest UCT-value (*U*pper *C*onfidence bounds applied to *T*rees) [25] is chosen based on hypervolume. The UCT-value is composed of two terms. First the hypervolume of the Pareto front (Exploitation) and secondly a score based on the number of visits of this state in relation to the overall number of visits. The second term, also known as exploration term, takes high values for states with few visits. Exploitation and exploration is balanced by a factor. In the second MO-MCTS phase, an action is added to the selected node based on the mechanism described in [26]. The roll-out phase used the same selection mechanism to determine actions until a terminal state is reached. During backpropagation the Pareto front of each node is updated. These four phases are repeated until the iteration limit is reached.

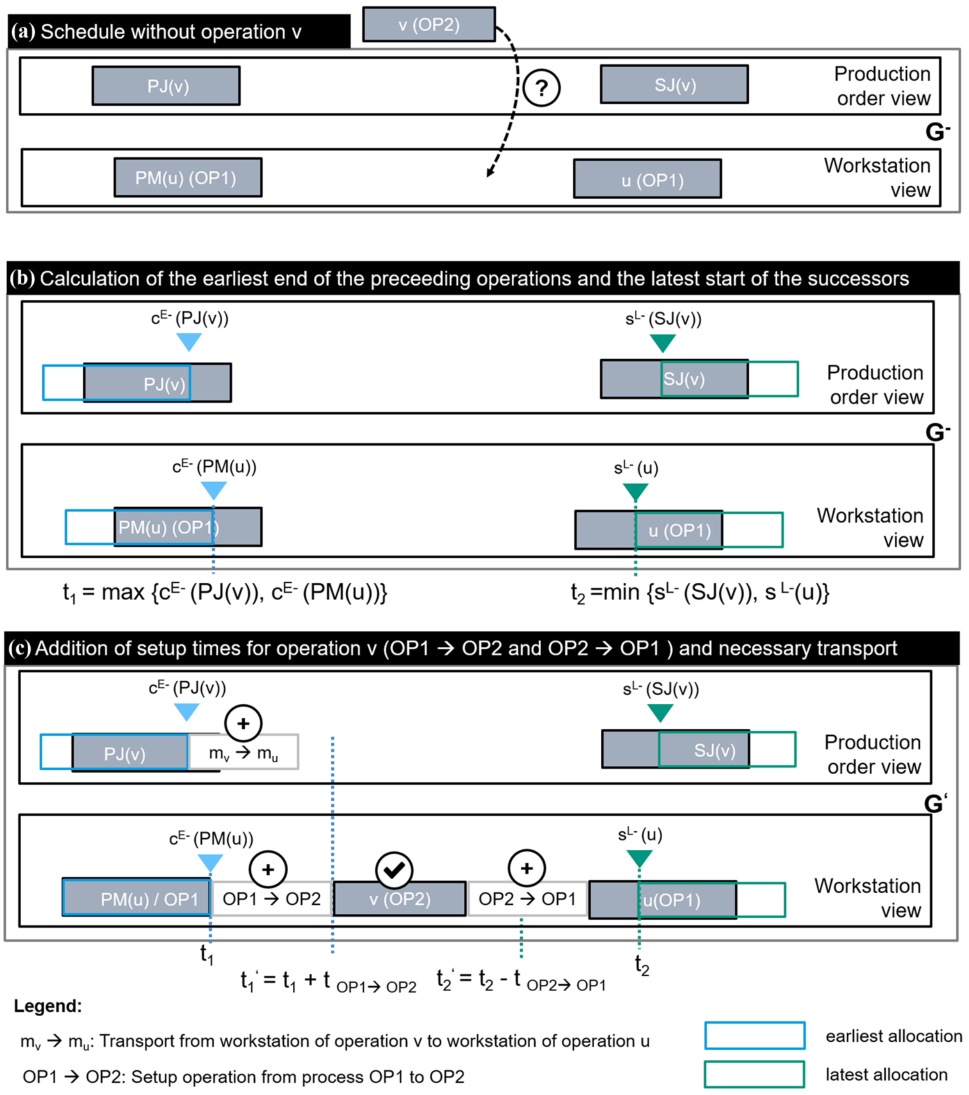
3.2 Local search post optimizer

Local search by moving one operation (LSONE) [27] combined with MO-MCTS [22] has achieved remarkable performance on the Kacem benchmark problems. However, LSONE does not consider transport and setup times since the original Kacem problem is only a simplified form of a matrix production. For LSONE, an operation is critical if the earliest starting point s^E and the latest starting point s^L are identical. A critical path consists only of critical operations and defines the makespan of the schedule. LSONE uses directed graphs to represent the scheduling problem. In the first step, the first critical operation is deleted from the Graph G to obtain the reduced graph G^-. Then, a new position for the previously deleted operation is determined that fulfills all time constraints. If an interval is found, the operation is inserted to obtain a complete schedule again. This is repeated until it is no longer possible to find a new interval [27]. LSONE has been applied by several authors in combination with MCTS [22]. The local search is usually executed after each roll-out. An important modification of LSONE has altered the original criteria applied to identify intervals to be more restrictive [22].

To accommodate transport and setup restrictions, changes have to be made regarding the definition of the critical path and the identification of suitable intervals. First, the definition of the critical path needs to be revised. The earliest starting time of operation v is given by $s^E(v)$. The earliest finishing time is equal to the cycle time of operation v at workstation w plus the earliest starting time $s^E(v)$. The latest start and finishing time are respectively defined as c^E and c^L. To determine the earliest starting time, it is necessary to verify both the earliest finishing time of the previous operation of this order $c^E(PJ(v))$ as well as the earliest completion time of the previous operation before operation v at the workstation w given by $c^E(PM(v))$. The maximum denotes the earliest starting time of operation v. Setup operations are regarded as previous operations on workstations and transport operations are modelled as previous operations of the order.

Secondly, the identification of possible intervals has to be adapted to account for additional or no-longer-needed setup or transport operations. The approach will be illustrated using the example in Fig. 1.

Figure 1 shows how the interval of operation v is determined. In this particular case, additional setup operations and a transport operation is needed. Starting point (a) is the reduced graph G^- without operation v of process $OP2$. To determine whether $v(OP2)$ can be placed between the operation $PM(u)(OP1)$ and $u(OP1)$, the earliest finishing time of the predecessors and the latest start of the successors have to be calculated. Both for the workstation as well as for the production order. The earliest starting time is restricted by the maximum of the earliest finishing time of the predecessor operation of the workstation $c^{E-}(PM(u))$ and of the production order $c^{E-}(PJ(v))$. Therefore, the earliest starting time is given by Eq. 1.



(a) Schedule without operation v
v (OP2)
PJ(v)
?
SJ(v)
Production order view
G-
PM(u) (OP1)
u (OP1)
Workstation view
(b) Calculation of the earliest end of the preceeding operations and the latest start of the successors
c^E- (PJ(v))
s^L- (SJ(v))
PJ(v)
SJ(v)
Production order view
G-
c^E- (PM(u))
s^L- (u)
PM(u) (OP1)
u (OP1)
Workstation view
t1 = max {c^E- (PJ(v)), c^E- (PM(u))}
t2 = min {s^L- (SJ(v)), s^L- (u)}
(c) Addition of setup times for operation v (OP1 → OP2 and OP2 → OP1) and necessary transport
c^E- (PJ(v))
s^L- (SJ(v))
PJ(v)
mv → mu
SJ(v)
Production order view
G'
c^E- (PM(u))
s^L- (u)
PM(u) / OP1
OP1 → OP2
v (OP2)
OP2 → OP1
u(OP1)
Workstation view
t1
t1' = t1 + t OP1→ OP2
t2' = t2 - t OP2→ OP1
t2
Legend:
mv → mu: Transport from workstation of operation v to workstation of operation u
OP1 → OP2: Setup operation from process OP1 to OP2
earliest allocation
latest allocation

Fig. 1 Illustrative example to demonstrate how to determine the interval for operation v

$$t_1 = max\{c^{E-}(PJ(v)), c^{E-}(PM(u))\} \tag{1}$$

The second restriction of the interval is given by the latest starting time of the predecessors, both on the workstation $s^{L-}(u)$ and of the production order $s^{L-}(SJ(v))$. Thus, the latest starting time is denoted by Eq. 2.

$$t_2 = min\{s^{L-}(SJ(v)), s^{L-}(u)\} \tag{2}$$

In the last step, additional setup and transportation operations need to be added to the graph. In the case illustrated, the production order needs to be transported t_{m_v,m_u} and an additional setup operation is needed before $t_{OP1 \mapsto OP2}$ and after the process $OP2$, $t_{OP2 \mapsto OP1}$. As a result t_1 is updated to Eq. 3. t_2 is changed respectively, see Eq. 4.

$$t_1' = max\{c^{E-}(PJ(v)) + t_{m_v,m_u}, c^{E-}(PM(u)) + t_{OP1 \mapsto OP2}\} \tag{3}$$

$$t_2' = min\{s^{L-}(SJ(v)), s^{L-}(u) - t_{OP2 \mapsto OP1}\} \tag{4}$$

The limits t_1' and t_2' are the borders of the assignable interval for the critical operation v.

3.3 Search strategy and interaction of local search and MO-MCTS

In this section two approaches will be presented to guide the local search. The first approach influences the order in which critical orders are considered and which assignable intervals are evaluated first. The second approach provides a means to undo changes that have not lead to the desired outcome.

3.3.1 Sorting of intervals and critical operations

Given a local search problem, there might exist several critical operations as well as several assignable intervals for each operation. The main idea is to use domain knowledge when choosing the intervals as well as operations to evaluate more promising alternatives first. This approach is in line with existing approaches in literature, e.g. preferring alternatives with shorter cycle times [28] or sorting the workstations according to the cycle time [29].

To sort critical operations, the following exemplary criteria has been tested. Other criteria can be applied as well:

- Duration.
- Earliest starting time.
- Ratio between setup times and cycle times.
- Delay.
- Waiting time.

To sort available assignable intervals similar criteria can be applied:

- Size of the interval.
- Earliest starting time.
- Ratio between setup times and cycle times.
- Setup state.
- Workload of the workstation.
- Idle time.

To save time by eliminating setup times, it is advisable to first test critical operations which belong to orders with a high ratio of setup time to cycle time and to evaluate assignable intervals with the required setup state.

3.3.2 Reallocation

In case of several assignable intervals for a critical operation, the choice of the assignable interval can be influenced by sorting, see Sect. 3.3.1. An alternative approach is to test multiple intervals in one iteration before selecting an allocation. To evaluate the resulting solution, the objective value of the resulting schedule can be compared to the initial objective value of the original schedule. To accept a solution, both static as well as dynamic thresholds are valid approaches. In case of local optima, sometimes a slight deterioration of the solution has to be accepted in one iteration to be able to leave the local optimum. The following criteria have been evaluated:

- Constant tolerance.
- Linear declining tolerance depending on the iteration count.
- Non-linear decline by Simulated annealing where the iteration count is used as temperature substitute.

3.3.3 Interaction of local search and base optimizer

In this section, the interaction between MO-MCTS as base-optimizer and local search are discussed. In general, local search can be ran on any complete schedule. Accounting for the fact that only minor improvement is to be expected by a neighborhood search, a pre-selection of the solutions to execute local search on is advisable. In literature [19, 22, 23] local search is executed on rollout states. To improve the efficiency of the entire optimizer, the following strategies have been developed:

- Rollout sampling: execution of local search after each rollout
- Intermediate Pareto states: This approach interrupts the execution of MO-MCTS after a fixed number of iterations or a fixed time budget and executes local search on a set of Pareto states.
- Final Pareto states: this approach executes local search on a set of Pareto states after the MO-MCTS execution has finished.

4 Results

This section discusses the effects of the modification of LSONE and assess the performance by applying the scheduler to a benchmark problem and an industrial use case. First, the modifications of LSONE to take setup and transportation into account are evaluated. Secondly, the effects of sorting of intervals and critical operations are assessed. Thirdly, the reallocation mechanism to test several moves in one iteration is discussed. To compare the scheduler to other approaches, it is applied to the Kacem benchmark problem. Lastly, the interaction of the scheduler and the production is demonstrated using an industrial application.

4.1 Use case

To evaluate the effects of the different approaches, the matrix production assembly of a supplier for electric drives for industrial applications has been taken as basis. The assembly process starts with the gearbox, followed by oil injection, motor assembly, testing and painting. In the real production system the gearbox assembly has to take place at a dedicated assembly station to simplify logistics. To represent a matrix production the assumption has been made that the logistics system of the gearbox assembly has been improved leading to a free choice of the assembly workstation. The cycle, setup and transport times are based on estimations from the planning phase [26].

4.2 Influence of LSONE modifications

LSONE has been modified compared to the original version [27] to account for transport and setup operations. To evaluate the effects of the modifications, the hypervolume and the equally-weighted objective value for delay, throughput time, makespan, total workload and maximum workload are compared.

Figure 2 shows the boxplots of 10 executions. The plot shows an improvement of the hybrid approach compared to the stand-alone optimizer in terms of hypervolume and equally-weighted objective values. The modified LSONE achieves superior results. The original LSONE criteria does not take into account setup and transportation times leading to the wrong identification of assignable intervals. It is worth noticing that the variance of the solution increases in both hybrid approaches. Compared to MO-MCTS, local search is less targeted and, thus, the solutions identified in the neighborhood of the starting point vary also in quality.

4.3 Sorting of critical operations and intervals

In this section, the influence of priority rules to sort critical operations and intervals are evaluated. Two different sorting schemes are discussed. First, the rules for sorting the critical operations and critical intervals are selected randomly. Secondly, matching rules for sorting are selected, e.g. if operations are prioritized according to the setup times then the equivalent rule is applied to sort the intervals.

Figure 3 shows the result of no sorting and the aforementioned sorting schemes regarding hypervolume and regarding the ratio of moves that lead to an improvement. It is evident, that sorting improves both hypervolume and the ratio of moves that lead to an improvement in terms of solution quality. Random sorting results in a comparable

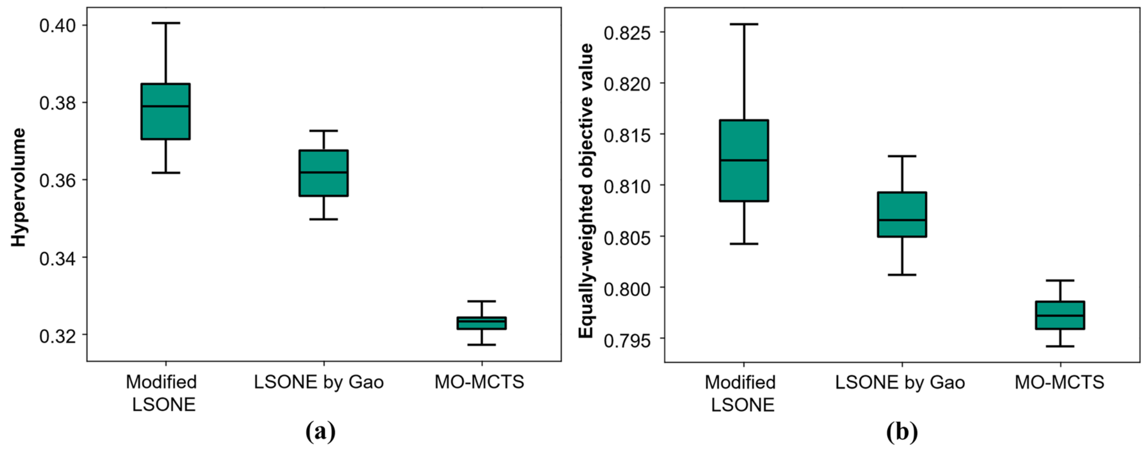


Fig. 2 Hypervolume (**a**) and equally-weighted objective value (**b**) for hybrid MO-MCTS with LSONE [27], modified LSONE and MO-MCTS as stand-alone optimizer

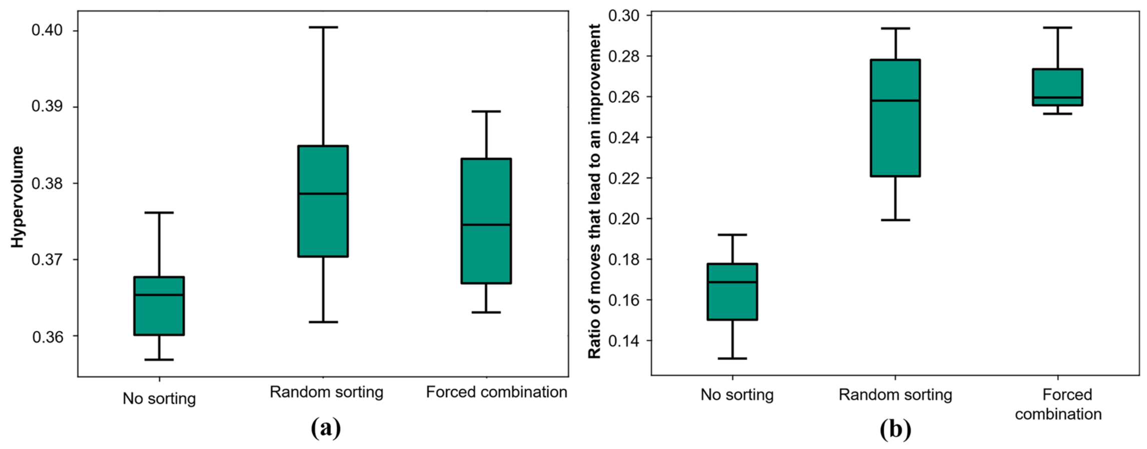


Fig. 3 Hypervolume (**a**) and ratio of moves (**b**) that leads to an improvement for different sorting schemes

performance as forced combinations. Random sorting even outperforms the forced combinations in rare cases. Regarding the success rate however, forced combinations lead to a more stable ratio of successful moves. The results suggest that forced combinations are a good approach in most local search situations but at the same time there remain many cases where additional effects across multiple machines and orders are not captured by these simple rules.

4.4 Reallocation

Before discussing different limits for reallocation, the effects of different definitions of an acceptable solutions are evaluated.

Figure 4 shows positive as well as negative constant tolerance values. Negative values mean that a decrease of the solution quality is permitted. It is clearly visible, that a tolerance of zero, thus, the acceptance of any solution that is at least as good as the previous solution leads to the highest hypervolume. Being to demanding (positive values) means that in case there are only marginal improvements possible no solution will be accepted. Regarding dynamic tolerance values, both simulated annealing with different starting values and linear functions have been tested. Both approaches show a comparable performance which is however inferior to the results of a constant tolerance level of zero. Therefore, a constant tolerance level of zero will be adopted for all further experiments.

The remainder of this section discusses the effects of a shallow compared to a thorough search. Three different influence factors are varied. First, the number of Pareto points to be considered as starting point for local search. The Pareto points are sorted by their equally-weighted objective value. The second influence factor is the total number of local search iterations to be performed. A value of 5 means that five different moves are performed. Each new move starting from the schedule resulting from the previous move. Lastly, the number of reassignments per iteration is defined. This value defines how many reassignments in one iteration starting from the same solution are conducted before the best one is selected. This variable determines the search thoroughness.

Figure 5 shows the result for the different combinations. The comparison of the extreme points (All, 1×5) and (5, 7 $\times$ all) reveals that a thorough search with seven iterations and maximum reassignment leads to better results compared to a shallow search. Thus, a thorough search on only 5 Pareto points leads to better results than a search with only one iteration on all Pareto points. Regarding the solution quality, a thorough search on fewer points is recommended. To compare the execution time, configurations which lead to the same hypervolume can be compared. Looking at the execution time, it is evident, that a thorough search not only leads to better results but also to a shorter execution time. The points (20, 3×10) and (10, $5 \times$ all) have almost an identical hypervolume but the thorough search is 24% faster. Hence, a thorough search favoring iterations and reallocations is recommended.

4.5 Interaction with MO-MCTS

To demonstrate the different effects of local search depending on the quality of the base optimizer, the experiments have been conducted with different numbers of simultaneously considered orders by MO-MCTS. This parameter is a complexity driver and has a significant impact on performance [26].

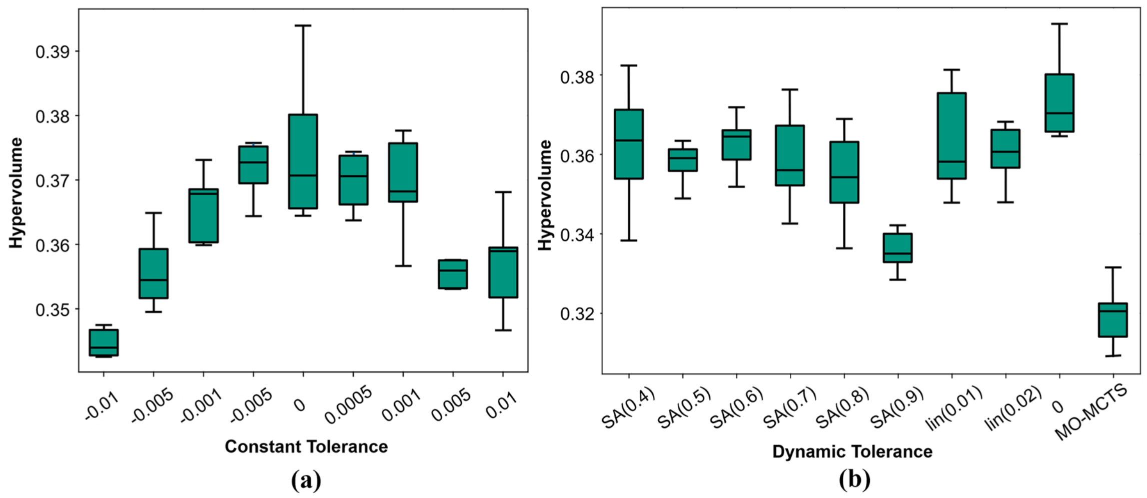


Fig. 4 Effect of different values for accepting solutions on the hypervolume

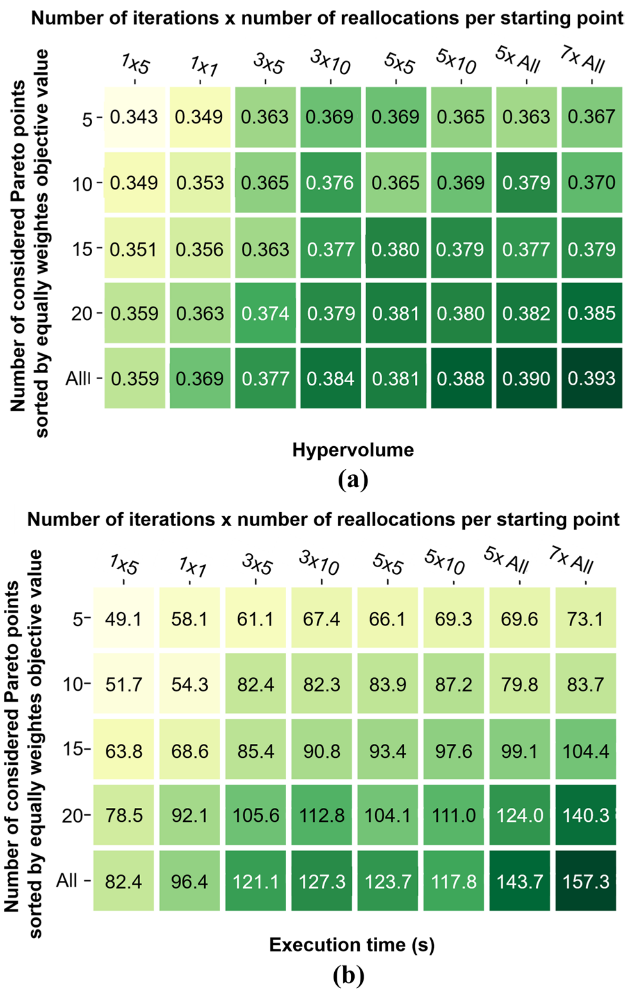


Fig. 5 Hypervolume (**a**) and execution time (**b**) depending on the number of iterations and number of reallocations and the number of Pareto points considered. The darkness of the colour gradient highlights large hypervolumes or longer execution times

Figure 6 shows the results in terms of hypervolume for MO-MCTS alone, the three interaction strategies of local search combined with MO-MCTS and a combination of all three depending on the number of simultaneously considered orders. Under regular circumstances, this parameter would be determined autonomously [26], for the purpose of demonstrating the varying effects of local search fixed numbers have been used in this experiment. The first observation is that the effect of local search is more pronounced when the MO-MCTS is not properly configured. For approximately seven orders, the improvement of the hybrid scheduler compared to MO-MCTS alone is less important. Intermediate sampling, here every 80 iterations, and final Pareto states sampling leads to very comparable results as the two curves demonstrate. Rollout sampling with a probability of 0.1 leads to improved results especially for non-ideal parameters of MO-MCTS. This is likely due to the fact, that taking random terminal states as a starting point for local search, increases the variability of the starting solutions. The combination of all three schemes results in a slightly better hypervolume for a well-tuned MO-MCTS but lower performance on the edges.

Overall, the experiment shows that in conjunction with a well-configured MO-MCTS all schemes lead to improvements. Rollout sampling slightly dominates the other alternatives at this point. However, final Pareto states sampling reduces the overall complexity of the scheduling system significantly as the Pareto points are handed over the local search at the end of the run and there is no effort for merging and restarting MO-MCTS required. If the interaction on both systems has been solved however, rollout sampling is recommended.

4.6 Application to Kacem benchmark

The Kacem benchmark problem is a widely-used scheduling problem in literature. Even though the Kacem benchmark lags some important characteristics of a matrix production such as transport and setup times, it is nonetheless a valuable means to compare different approaches. Besides the aforementioned shortcomings, the Kacem benchmark does neither provide alternative process sequences nor due dates. Additionally it has a strong focus on resource-related objectives (total workload and maximum workload) and on the makespan as overall performance indicator. Regarding the matrix production, the Kacem benchmark can be modelled as a simplified matrix with setup and transport times being zero and due dates in the far future.

Table 1 shows the results of relevant approaches on the five different instances of the Kacem benchmark. The approaches from literature are tailored to solve the Kacem benchmark whereas no modifications have been made to the scheduling approach presented in this paper.

The last line shows the results obtained by the given approach. 15 of the 18 known Pareto points can be identified in a 120 seconds run on an ordinary dual core laptop. The Pareto points of the largest instance are missed by one unit ((12, 91, 11) and (13, 93, 10)). Taking (12, 92, 11) as a starting point, local search can improve the result to obtain (11, 93, 11).

The application to the Kacem benchmark shows that the given approach leads to good results even compared with approaches that are tailored to the Kacem problem structure and objectives.

4.7 Industrial use case

To illustrate the reactive behavior of the scheduler, it is applied to an industrial use. The real production system is represented by an discrete-event simulation model.

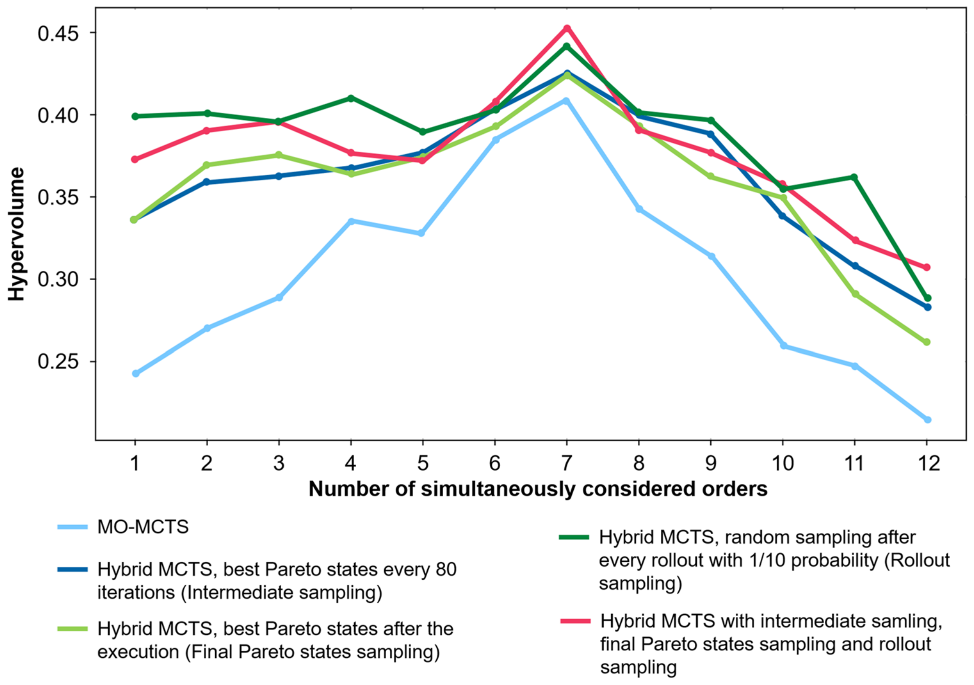


Fig. 6 Hypervolume depending on the number of simultaneously considered orders and the interaction of local search with MO-MCTS

Table 1 Results on the Kacem benchmark [30] not found

	(4x5)				(10x7)			(8x8)					(10x10)				(15x10)	
Makespan*	11	11	12	13	11	11	12	14	15	15	16	16	7	7	8	8	11	11
Total workload*	32	34	32	33	61	62	60	77	75	81	73	77	42	43	42	41	91	93
Maximum workload*	10	9	8	7	11	10	12	12	12	11	13	11	6	5	5	7	11	10
Multi-objective GA [12]	●	●	●	○	-	-	-	○	●	●	●	○	●	○	●	●	●	○
Simple evolutionary algorithm [31]	●	●	●	●	●	●	●	●	●	○	●	●	●	●	●	●	●	●
Mememic algorithm [13]	●	●	●	●	●	●	●	●	●	○	●	●	●	●	●	●	●	●
MO-MCTS + LS [22]	●	●	●	●	●	●	●	●	●	○	●	●	●	●	●	●	●	●
MO-MCTS + LS [23]	●	●	●	●	●	●	●	●	●	○	●	●	●	●	●	●	●	●
MO-MCTS [26]	●	●	●	●	●	●	●	●	●	○	●	●	●	●	●	●	○	○
MO-MCTS + LS	●	●	●	●	●	●	●	●	●	○	●	●	●	●	●	●	○	○

not found: ○, found: ●, unknown: -, optimum:*

The production system consists of six gear assembly workstations (GA1–GA6), followed by oil filling, motor assembly (MA1–MA6), testing (T1–T4) and painting as introduced in Sect. 4.1. This production system also serves as basis for the generic system used to evaluate the scheduler in the preceding sections. In this real-life scenario, break-downs occur and operations can be delayed. To account for the inflexibility due to material transport, production orders are not allowed to be reallocated within the next 20 min, unless a break-down occurred. The hybrid scheduling system determines the best schedule and transfers it to the simulation. The simulation system returns the current state each time an operation has finished or should be finished, a new order arrives or a machine status changes. The hybrid scheduling system then verifies if the current state is still in line with the planning. If this is not the case, a new schedule is calculated based on the current situation.

Figure 7 shows the schedule at two different points in time. On the bottom line is the first schedule that has been generated at the beginning. The second line depicts a schedule of a later iteration after a break-down at GA5 occurred. The reassignment of the impacted operations to different workstations is clearly visible. GA3 demonstrates how the hybrid scheduler reacts to deviations within the 20 minutes corridor of reduced flexibility. In this scenario, local search improves makespan by 12.7% on average.

5 Conclusion and outlook

This paper addresses scheduling of a matrix production taking into account realistic restrictions such as transport and setup times. Building on the findings that hybrid MCTS schedulers have achieved remarkable results on the related but simpler Kacem benchmark problem, this paper contributes a hybrid MO-MCTS approach for matrix production. Core of this paper is the adaption of the local search approach LSONE to incorporate setup and transport as additional restrictions. The modifications of LSONE are analyzed in detail and the obtained scheduler is tested on the Kacem benchmark and on an industrial problem to illustrate

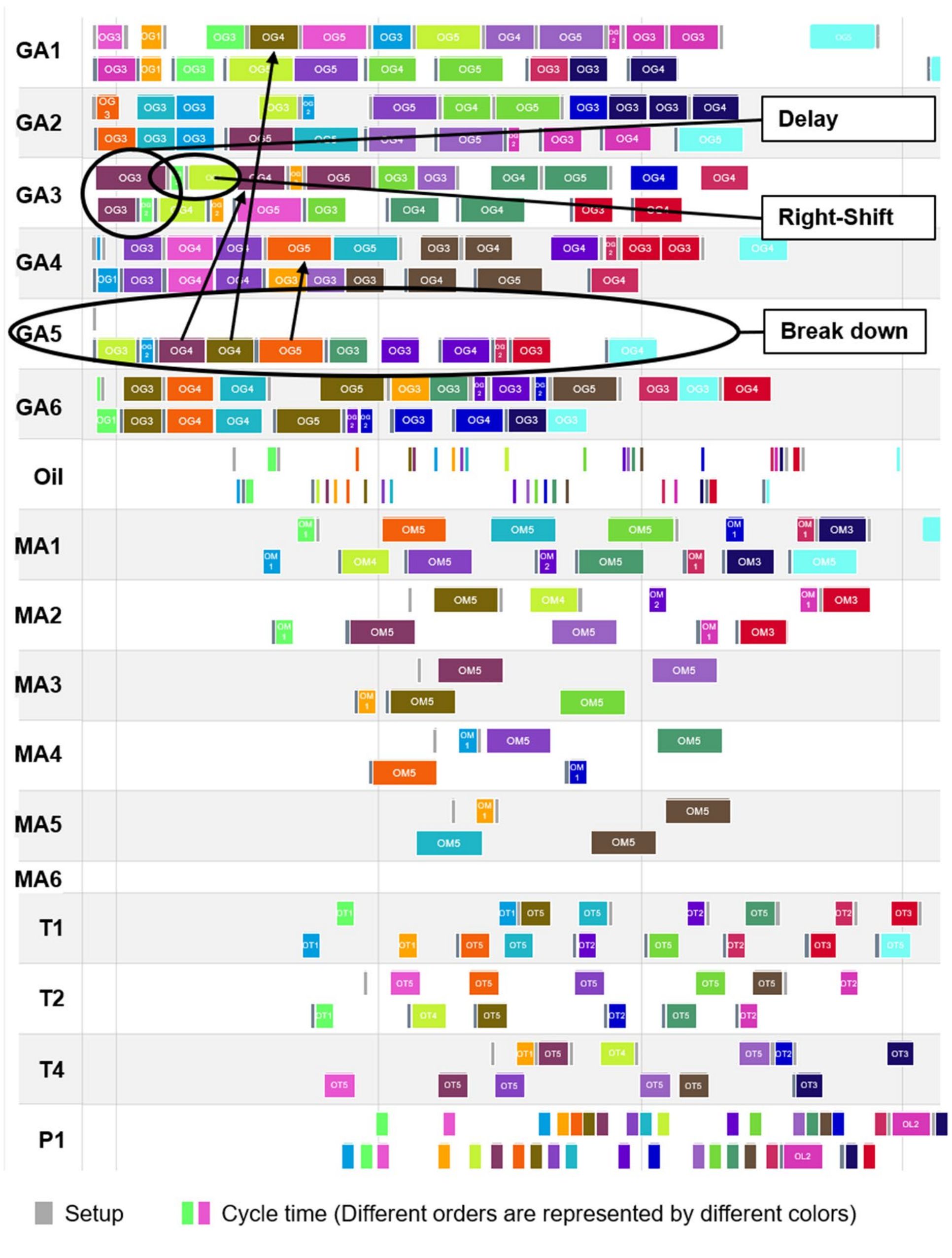


Fig. 7 Initial schedule and intermediate schedule for an industrial example in the context of break downs and delays

its behavior in the context of break-downs and delays. It can be shown that the hybrid scheduling system performs well on the benchmark and is able to react on disturbances in the industrial use case. A thorough evaluation of the scheduling approach in the context of an industrial production system cannot be provided and should be addressed in a separate study. Whilst the MO-MCTS base optimizer improves all given objectives, by design, the local search approach mainly reduces makespan. This weakness is worth addressing in future work by applying dedicated swap and shift strategies in order to target other objectives likewise. Regarding further research, different strategies of interacting with the production could be evaluated. In the current example, disturbances are not anticipated. An analysis of an appropriate strategy depending on the disturbances (break-downs, delays) would be beneficial to pave the way for a real application. Building on the finding, that sorting of intervals and critical operations based on domain knowledge improved the results, further research could focus on an adaptive approach to identify suitable problem-specific heuristics.

Acknowledgements Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)—LA 2351/51.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Hu SJ, Zhu X, Wang H, Koren Y (2008) Product variety and manufacturing complexity in assembly systems and supply chains. CIRP Ann 57(1):45–48. https://doi.org/10.1016/j.cirp.2008.03.138
2. Lanza G, Nyhuis P, Fisel J, Jacob A, Nielsen L, Schmidt M, Stricker N (2018) Wandlungsfähige, menschzentrierte Strukturen in Fabriken und Netzwerken der Industrie 4.0 (acatech Studie). Technical report. Deutsche Akademie der Technikwissenschaften, acatech, München
3. Greschke PI (2016) Matrix-Produktion Als Konzept Einer Taktunabhängigen Fließfertigung, Universität Braunschweig, Dissertation. Vulkan Verlag, Essen
4. Hüttemann G (2021) Model-based: a priori analysis of line-less mobile assembly systems, RWTH Aachen University, Dissertation. Apprimus Verlag, Aachen
5. Schönemann M, Herrmann C, Greschke P, Thiede S (2015) Simulation of matrix-structured manufacturing systems. J Manuf Syst 37:104–112. https://doi.org/10.1016/J.JMSY.2015.09.002
6. Pinedo ML (2016) Scheduling: theory, algorithms, and systems. Springer, New York
7. Garey MR, Johnson DS, Sethi R (1976) The complexity of flowshop and jobshop scheduling. Math Oper Res 1(2):117–129
8. Özgüven C, Özbakır L, Yavuz Y (2010) Mathematical models for job-shop scheduling problems with routing and process plan flexibility. Appl Math Model 34(6):1539–1548. https://doi.org/10.1016/J.APM.2009.09.002
9. Ouelhadj D, Petrovic S (2008) A survey of dynamic scheduling in manufacturing systems. J Sched 12(4):417–431. https://doi.org/10.1007/s10951-008-0090-8
10. Uzsoy R, Lee CY, Martin-Vega LA (1994) A review of production planning and scheduling models in the semiconductor industry part II: shop-floor control. IIE Trans (Institute of Industrial Engineers) 26(5):44–55. https://doi.org/10.1080/07408179408966627
11. Scholz-Reiter B, Görges M, Philipp T (2009) Autonomously controlled production systems-Influence of autonomous control level on logistic performance. CIRP Ann 58(1):395–398. https://doi.org/10.1016/J.CIRP.2009.03.011
12. 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(5–8):757–767. https://doi.org/10.1007/s00170-010-2642-2
13. Chiang T-C, Lin H-J (2012) Flexible job shop scheduling using a multiobjective memetic algorithm. In: Huang DS, Gan Y, Bevilacqua V, Figueroa JC (eds) Advanced intelligent computing, 7th international conference, ICIC 2011, 11–14. August 2011, Zhengzhou. Springer, Berlin, pp 49–56
14. Zhen H-L, Wang Z, Li X, Zhang Q, Yuan M, Zeng J (2021) Accelerate the optimization of large-scale manufacturing planning using game theory. Complex Intell Syst. https://doi.org/10.1007/s40747-021-00352-7
15. Nie L, Wang X, Pan F (2019) A game-theory approach based on genetic algorithm for flexible job shop scheduling problem. J Phys Conf Ser 1187(3):32095. https://doi.org/10.1088/1742-6596/1187/3/032095
16. Silver D, Huang A, Maddison CJ, Guez A, Sifre L, van den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M, Dieleman S, Grewe D, Nham J, Kalchbrenner N, Sutskever I, Lillicrap T, Leach M, Kavukcuoglu K, Graepel T, Hassabis D (2016) Mastering the game of Go with deep neural networks and tree search. Nature 529(7587):484–489
17. Browne C, Powley E, Whitehouse D, Lucas S, Cowling P, Rohlfshagen P, Tavener S, Liebana D, Samothrakis S, Colton S (2012) A survey of Monte Carlo tree search methods. IEEE Trans Comput Intell AI Games 4(1):1–49
18. Lubosch M, Kunath M, Winkler H (2018) Industrial scheduling with Monte Carlo tree search and machine learning. In: Wang L (ed) Procedia CIRP, 51st CIRP conference on manufacturing systems (CMS), 16–18. May 2018, Stockholm. Elsevier B.V., Amsterdam, pp 1283–1287. https://doi.org/10.1016/J.PROCIR.2018.03.171
19. Lu C-L, Chiu S-Y, Wu J, Chao L-P (2016) Dynamic Monte-Carlo tree search algorithm for multi-objective flexible job-shop scheduling problem. Appl Math 10(4):1531–1539
20. Domschke W, Drexl A, Klein R, Scholl A (2015) Einführung in operations research. Springer, Berlin. https://doi.org/10.1007/978-3-662-48216-2
21. Blazewicz J, Domschke W, Pesch E (1996) The job shop scheduling problem: conventional and new solution techniques. Eur J Oper Res 93(1):1–33
22. Wu T, Wu I, Liang C (2013) Multi-objective Flexible job shop scheduling problem based on Monte-Carlo tree search. In: Li

D (ed.) TAAI '13: proceedings of the 2013 conference on technologies and applications of artificial intelligence, 2013 conference on technologies and applications of artificial intelligence, 6–8 Dez 2013, Taipei. IEEE, Washington, pp 73–78. https://doi.org/10.1109/TAAI.2013.27

23. Chou JJ, Liang CC, Wu HC, Wu IC, Wu TY (2015) A new MCTS-based algorithm for multi-objective flexible job shop scheduling problem. In: Wang H-C, Chen R-M, Chang B-R (eds) Proceedings of a meeting held 20–22 November 2015, Tainan, 2015 conference on technologies and applications of artificial intelligence (TAAI 2015), 20–22 November 2015, Tainan. IEEE, New York, pp 136–141. https://doi.org/10.1109/TAAI.2015.7407061
24. Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans Evol Comput 3(4):257–271
25. Kocsis L, Szepesvári C (2006) Bandit based Monte-Carlo planning. In: Fürnkranz J, Scheffer T, Spiliopoulou M (eds) Machine learning: ECML 2006, 17th European conference on machine learning, 18–22 2006, Berlin. Springer, Berlin, pp 282–293
26. Stricker N, Kuhnle A, Hofmann C, Deininger P (2021) Self-adjusting multi-objective scheduling based on Monte Carlo tree search for matrix production assembly systems. CIRP Ann 70(1):381–384
27. Gao J, Sun L, Gen M (2008) A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. Comput Oper Res 35(9):2892–2907. https://doi.org/10.1016/j.cor.2007.01.001
28. Ho NB, Tay JC (2008) Solving multiple-objective flexible job shop problems by evolution and local search. IEEE Trans Syst Man Cybern Part C (Applications and Reviews) 38(5):674–685. https://doi.org/10.1109/TSMCC.2008.923888
29. Xia W, Wu Z (2005) An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. Comput Ind Eng 48(2):409–425. https://doi.org/10.1016/J.CIE.2005.01.018
30. Kacem I, Hammadi S, Borne P (2002) Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic. Math Comput Simul 60(3–5):245–276. https://doi.org/10.1016/S0378-4754(02)00019-8
31. Chiang T-C, Lin H-J (2013) A simple and effective evolutionary algorithm for multiobjective flexible job shop scheduling. Int J Prod Econ 141(1):87–98. https://doi.org/10.1016/j.ijpe.2012.03.034