Agent-based, hybrid control architecture for optimized and flexible production scheduling and control in remanufacturing

Sebastian Groß M. Eng[1,2]*, Prof. Dr.-Ing. Wolfgang Gerke[1§], Prof. Dr. Peter Plapper [2§]

[1] Department of Robotics and Remanufacturing, Tirer University of Applied Science, Environmental Campus Birkenfeld, Campusallee, 55761 Birkenfeld, GER
[2] Department of Intelligent Robotics, University of Luxembourg Rue Richard Coudenhove-Kalergi, L-1359, LUX

*These authors contributed equally to this work
[§]Corresponding author

Email addresses:
SG: s.gross@umwelt-campus.de
WG: w.gerke@umwelt-campus.de
PP: peterplapper@uni.lu

Abstract

Motivated by high ecological and economical potentials and driven by new laws, remanufacturing is receiving increasing attention as a process that puts used products into "as good as new or better" condition. Within this process, there are many challenges, which are unknown from manufacturing, such as the uncertainties resulting from unknown conditions of the used products. This places special demands on the control of the remanufacturing system (RS). To handle these uncertainties an agent-based hybrid control architecture comprising centralized and decentralized components is presented. In the former, the scheduling takes place including the consideration of the use of automated guided vehicles (AGV) to realize of flexible material handling within the RS. The scheduling of machines and AGVs is thereby considered simultaneously and not separately, as it is the case in currently available control systems. For the optimization of this simultaneous scheduling Constraint Programming (CP) is used. In the decentralized component, all participants within the RS will be networked as a cyber-physical system and controlled by respective agents. These agents can communicate with each other in order to find solutions. The architecture is implemented as a multi-agent system.

Simulation results, using benchmark instances, show that simultaneous scheduling results in a 19.7% reduction of the makespan. Furthermore, the CP-based approach delivers the best results, compared to other approaches for simultaneous scheduling, which are also achieved in a significantly shorter computing time.

Keywords

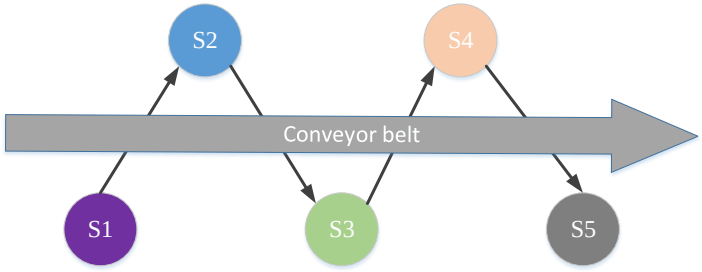
Simultaneous scheduling, constraint programming, multi-agent systems, automated guided vehicles, production planning and control, flexible remanufacturing systems

Introduction

In the domain of remanufacturing many challenges occur, which are special to remanufacturing and not known from manufacturing. One of these challenges are the unknown conditions of the used products. The product condition can be very different from product to product, depending on the use, which means that every product can take a different route through the RS because different remanufacturing processes are required. Like shown in Fig 1 even the same product, here *Product A*, can take a different route through the remanufacturing system based on different product condition and therefor different required remanufacturing process steps. This leads to stochastic routings of the products through the RS, which requires a flexible material handling system on the shop floor like Automated Guided Vehicles (AGVs). AGVs are considered as one of the most important enablers of flexible material handling on the shop floor [1]. Fig 1 shows the difference between a traditional manufacturing line and a flexible remanufacturing system (FRS). In the former, every product of the same type goes through the same workstations (*S1-Sx*), which are connected by a conveyor belts, in the same order. In the FRS on the other hand the products take different routes through the available workstations, which are connected through AGVs. This makes the production planning and control (PPC) of remanufacturing systems difficult. One reason for this is that the process steps, required remanufacturing the product at hand, are only known after an initial inspection. Even after the initial

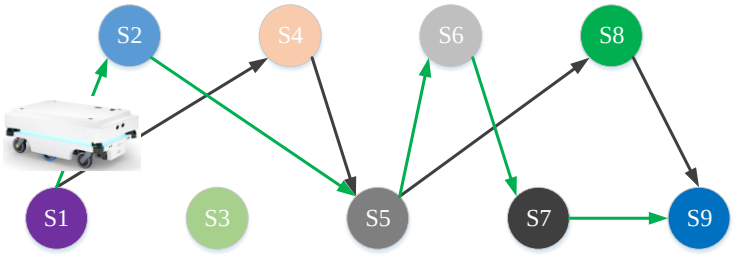
inspection, the product condition is not always completely known and therefore unexpected events may occur during the execution of the different remanufacturing processes.

Therefor the control of the FRA has to be flexible in order to be able to react adequately to the unknown condition of a product as well as to unexpected internal and external events. The associated scheduling and control of the available AGVs needs to be taken into consideration in the PPC as well. However, the PPC systems currently available often do not adequately reflect the complexity and volatility resulting from the above-mentioned circumstances [2]. In addition, the PPC systems usually also do not integrate the management of AGVs. These are managed via external fleet management software. A holistic approach, especially in the area of simultaneous scheduling of machines and AGVs (SSMA), can lead to optimization potentials.



Single route through the manufacturing line:
Product A: S1 -> S2 -> S3 -> S4 -> S5

Traditional manufacturing line



Multiple routes through the remanufacturing system for the same product:
Product A1: S1 -> S2 -> S5 -> S6 -> S7 -> S9
Product A2: S1 -> S4 -> S5 -> S8 -> S9

Flexible remanufacturing system

Fig 1: Comparison of a traditional production line with a flexible remanufacturing system and the use of AGVs as material handling systems.

A promising approach to enable the required flexibility of the control system and to control the dynamics contained within the FRS is the use of multi-agent technology [3], [4]. This approach defines the various resources in the (re-)manufacturing system as intelligent agents that negotiate with each other in order to perform dynamic reconfigurations and to achieve high flexibility [5]–[10].

The multi-agent technology, which is already around for several decades, is a field of artificial intelligence and close to distributed problem solving [11]. A multi-agent system (MAS) consist of several agents with different abilities that collaborate with each other in order to complete certain tasks and to achieve certain goals [12]. An agent can be a software-technical image of a physical unit as well as a pure virtual unit and is defined as a computer system which is located in a certain environment in which it can act autonomously in order to achieve its goals [13]. Within a MAS several agents interact and negotiate with each other in order to find an optimal solution. They can either pursue different and contradictory goals, or consequently work together to achieve a common, superior goal.

In this paper an agent-based, hybrid control architecture is proposed with the goal to optimize the scheduling of the FRS and to handle the stochastic within the FRS. The hybrid architecture consists of a centralized, as well as a decentralized component. In the former, the scheduling takes place including the consideration of the use of automated guided vehicles (AGV) for flexible material handling. The scheduling of machines and AGVs is thereby considered simultaneously and not separately, as it is the case in currently available control software. For the optimization of this simultaneous scheduling Constraint Programming (CP) is used. In the decentralized component, all participants within the FRS will be networked as a cyber-physical system and controlled by respective agents. These agents can communicate with each other in order to find solutions. The architecture is implemented as a MAS consisting of several agents and agent types.

This architecture allows it to master the control of the FRS with the required flexibility for the different routes of the products through the FRS.

The approach will be tested and validated through simulation using benchmark instances as well as the implementation of the proposed approach to control a demonstrator.

This paper is structured as follows: in the second part the state-of-the-art regarding the PPC in the domain of remanufacturing, MAS in PPC, fleet management for AGVs and SSMA is presented; in the third part the architecture and methodology of the proposed agent-based, hybrid control architecture as well as the used CP approach regarding the SSMA will be represented; in the fourth part simulation results are presented to show the superiority of the proposed approach and in the last part a conclusion will be given and future work will be described.

State-of-the-art

PPC in remanufacturing

The goal of PPC is the optimization of the entire production system. PPC consist of the three parts, planning, scheduling and control. The requirements for PPC-System depend on the type of production system. In the manufacturing sector, for example, special solutions for job shop, flexible job shop [14]–[21] or re-entrant manufacturing systems are necessary and examined by researchers [7], [8], [10], [12], [22]. In the field of remanufacturing there are however particular characteristics and challenges for the PPC which are not known from manufacturing activities [23]–[28]. Some of these challenges are:

- Unknown product condition of the used-products due to different stress during their usage
- Unknown inspection results of the used-products
- Unknown condition of the disassembled parts
- Varying processing times
- Balance between customer demand and return of used-products
- Stochastic routings for materials and products
- Unknown arrival time and quantity of used-products

These characteristics, unique to the domain of remanufacturing, require a different approach for PPC then in the field of traditional manufacturing systems [29]. Especially the uncertainties and stochastic routings in remanufacturing need to be taken into consideration for the PPC.

Research activities in the field of PPC for remanufacturing often focus on the disassembly planning and scheduling. Junior and Filho [30] proposed a stochastic dynamic programming model to master the production scheduling for disassembly in the remanufacturing environment. The proposed model was tested and validated on a real case of automotive clutch remanufacturing. Franz [31] proposed a mixed-integer-program (MIP) to solve the disassembly planning and scheduling. The MIP is based on disassembly process graph, which represents parallel and alternative operations. To demonstrate the use of the approach in a real disassembly context an industrial case study is presented. Jungbluth et al. [32] proposed an intelligent robot assistant for the non-destructive disassembly in remanufacturing, that is controlled using a MAS, whereby the tools, robots and workers used are represented by agents. The assistance system carries out the disassembly planning on the basis of an extended CAD product model, divides the tasks between man and robot and automatically adjusts the disassembly plan if unexpected events occur. In addition, the RAS controls the collaborative robot and synchronizes the work between human and robot. To approach was successfully test and validated on a demonstrator for the disassembly of electric gear-drives.

Current research activities are also carried out in the area of production scheduling in remanufacturing. He [33] proposed an hybrid algorithm based on a backpropagation neural network and a genetic algorithm for the simulation and optimization of remanufacturing production scheduling under uncertainties. The approach was validated through simulation. Kim et al. [34] proposed a mixed integer programming model for the problem representation and a scheduling approach based on priority rules. Validation of the approach was done through simulation on different test instances and a remanufacturing model factory. Wen et al. [35] developed a hybrid algorithm using bi-random simulation technique, neural network and genetic algorithms to perform and optimize the production planning and control in a remanufacturing system with uncertainties. The approach is tested and validated through a case study.

However all of the above mentioned approaches do not take the SSMA for FRS into account.

Multi-agent systems in Production planning and control

With regard to the use of MAS in the area of PPC, there are many different research focuses. Leusin describes a MAS approach [10] for real-time production control in workshop production. Luo et al. presents an approach where the complete manufacturing execution system (MES) is modelled by a MAS to realize a flexible design of the existing production process [36]. Merdan et al. have developed a distributed, intelligent control system based on Automation Agents, which enables an improvement in the response to failures or overloading of components [37]. Vojdani et al. use an agent-based approach for detailed production planning to improve the calculation of delivery dates by using real-time production data in the simulation [38]. Lima et al. develop an agent-based PPC system that dynamically adapts to changes in the production system [39]. Scholz-Reiter et al. uses the modelling and simulation of production based on agents for the self-control of logistic processes. The product finds its way through the production system itself [40]. He et al. present a hierarchical, agent-based tendering mechanism, which is specifically designed for make-to-order production. This operates within defined framework conditions and ensures that resources are used in a self-organized and cost-efficient manner to fulfil customer orders [41].

Fleet Manager for AGVs

AGVs are used as flexible material handling systems in the (re-)manufacturing industry and are able to move products and materials without pre-defined routes. Commercial available AGVs provide different approaches of self-guided navigation in order to find a collision-free path between workstations. If several AGVs exist on the shop floor, fleet management systems are used for the scheduling and supervision of the AGV fleet. Currently available fleet managers just focus on the localisation and navigation of the AGVs. To minimize the transport time of materials just the optimization of routes and the allocation of the best AGV for the task at hand is taken into consideration.

Different fleet managers are developed and available from various manufacturers of AGVs. *Mobile Industrial Robots* for example provides the *MiRFleet* [42], which allows the collision free routing of various robots. The system also provides the ability to assign tasks with priority rules. Furthermore, the system monitors the battery charge levels of the AGVs and manages automatically the charging processes. The fleet management systems from *KUKA AG*, the *KUKA.NavigationSolution* [43], and the *AGV Manager* from *BA Systèmes* [44] attempt to reduce the overall travel time by taking the production environment, the traffic and the required target location into consideration. These systems provide job scheduling and real-time routing as outputs. *DEMATIC's E'tricc* AGV fleet manager [45] selects the AGVs to the tasks by analysing the workflow as well as re-evaluating assignments. An analysis of historical travel routing and operation data is implemented in the *AGV Supervision system* from *Sidel* [46][43] and the *SGV Manager* from *JBT* [47] in order to optimise the performance of the AGV fleet in industrial environment. The *Vehicle Manager* from *savant automation* [48] is able to process inputs from network computer systems, discrete I/O, PLC network etc. in order to assign the available AGVs to tasks. Furthermore, the Vehicle Manager achieves and considers historical data.

Besides the listed commercially available software solutions, the subject of fleet management for AGVs is also the subject of some current research projects. Srivastava et al. [49] for example present an agent-based approach for operation control of an AGV fleet with the goal to find a collision-free and time optimised path in the AGV path networks. The simulation functions for evaluating different scenarios within this fleet manager provide an efficient and a validated solution for AGVs running in complex flow networks. Regarding the fleet management of multiple AGVs in industrial warehouses Cardarelli et al. [50] proposed cooperative cloud robotics architecture. Through cooperative data fusion from various sensor systems, a continually updated global live view of the environment was achieved. The goal was to provide a collision-free path if unexpected obstacles occur in the environment. The methodology was successfully validated in a real industrial environment. Yao et al. [51] provides a Smart AGV Management System to optimize the scheduling of AGVs in a manufacturing process. The proposed approach uses the combination of real-time data analysis and a digital twin model to optimize the schedule. For a proof of concept, the approach was successfully tested on demonstrator with a manual assembly station.

However, none of the listed, available software solutions as well none of the stated research projects provides the possibility regarding an integration for the SSMA to optimize the overall scheduling with the use of real-time production data [51]

Simultaneous scheduling for machines and AGVs

Traditionally, scheduling problems have considered machines as the only important resource, but this is no longer true as material handling in an FRS becomes more valuable and transport times contribute to machine downtime as machines have to wait for the next part to be machined. Extensive research has been

devoted to machine scheduling and vehicle scheduling independently, but the two problems are closely linked. Few have directed their research to the SSMA in an FRS. The scheduling in an FRS is conceptually similar to Job-Shop Scheduling Problems (JSSP), with the difference that in a JSSP material handling is not considered. The goal of the JSSP is to define the processing order for all the operations out of a set of jobs to a set of machines while minimizing the makespan. The machine to process a certain operation is predefined here. Sequence conditions between the operations of a job and the fact that one machine can only process one operation at one time has to be taken into consideration. In the JSSP every operation is processed on one specific machine, whereas in the extension of the JSSP the Flexible Job-Shop Scheduling Problem (FJSSP) every operation can be processed on at least one or more machines. The JSSP and the FJSSP are NP-hard problems [52] and studied by many researchers. An example of a JSSP is shown in Table 1. This JSSP consists of three jobs (rows within the table), each consisting of two to three operations (columns) and three available machines (M1, M2, M3). The values within the brackets after the respective machines correspond to the process time of the corresponding operation on this machine.

	Operations		
Jobs	1	2	2
Job 1	M1 (16)	M2 (8)	M3 (4)
Job 2	M3 (9)	M2 (9)	
Job 3	M2 (7)	M3 (10)	M1 (12)

Table 1: Example of an JSSP with three jobs, eight operations and three machines.

The solution to the JSSP problem is a production plan (schedule). An often used form of representing a schedule is the Gantt chart (see Fig 2). It shows the available machines on the ordinate axis and the time on the abscissa axis. The individual operations of the jobs are entered in the Gantt diagram in the line of the assigned machine with the respective start and end times as well as the corresponding process time. The second operation of the third job O_{32} is assigned to machine M3 in Fig 2. The start time of this operation is S_{32} and corresponding to the process time of P_{32} on this machine the end time is E_{32}. t_0 corresponds either to time $t_0 = 0$ in the initial scheduling or to the current time when a new execution of the scheduling is executed (rescheduling). The schedule in Fig 2 also represents a permitted schedule of the JSSP example from Table 1.

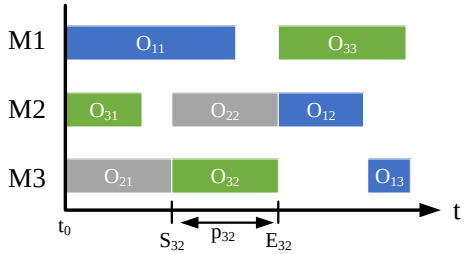


Fig 2: One possible schedule for the JSSP example in Table 1.

In the SSMA, all available machines are considered as well as all available AGVs, which have to transport the individual products between the machines. Only few research is done regarding the SSMA. In Table 2 the state-of-the-art regarding SSMA is represented. In the left column, the researchers are named and the corresponding article is linked. In the column *Method* the method used in the article to solve the simultaneous scheduling is named. The three right columns show whether in the corresponding article the investigation was performed in an FJSSP environment, with alternative machine, or in a JSSP environment, without such, or whether dynamics was taken into account. Dynamic events can be the arrival of new job, or the breakdown of a machine, respectively an AGV, for example. If the according case applies, the corresponding table field contains a "✓", otherwise an "O".

Article	Method	Dynamic	FJSSP	JSSP
Bilge and Ulusoy [53]	Non-linear mixed integer programming model	O	O	✓
Nageswararao et al. [54]	Binary Paricle Swarm Vehicle Heuristic Algorithm (BPSVHA)	O	O	✓
Erol et al. [55]	Multi-Agent System	O	O	✓
Mousavi et al. [56]	Hybrid algorithm consisting of Genetic Algorithm and Particle Swarm Optimization	O	O	✓
Chaudhry et al. [57]	Genetic Algortihm	O	O	✓

Fontes and Homayouni [58]	Mixed Integer linear Programming model	O	O	✓
Lacomme et al. [59]	Memetic Algorithm	O	O	✓
Faudi and Murata [60]	Binary Paricle Swarm Optimization	O	O	✓
Deroussi and Norre [61]	Iterative Local Search (ILS)	O	✓	O
Zhang et al. [20]	Hybrid algorithm with a combination of a Genetic Algorithm and a Tabu-Search	O	✓	O
Zhang et al. [62]	Hybrid algorithm with a combination of a Genetic Algorithm, a Shifting Bottleneck Procedure and a Tabu-Search	O	✓	O
Kumar et al. [63]	Hybrid Algorithm with a Differential Evolution algorithm a Vehicle Assignment Heuristic and a Machine Selection Heuristic	O	✓	O
Sahin et al. [64]	Multi-Agent System	✓	✓	O
Lin et al. [65]	Simulation-based optimization	✓	✓	O
Deroussi et al. [66]	Hybrid algorithm of Particle Swarm Optimization with a stochastic Local Search	O	✓	O
Nouri et al. [67]	Hybrid algorithm of a Genetic Algorithm and a Tabu-Search	O	✓	O
Homayouni and Ponto [68]	Mixed Integer Linear Programming model	O	✓	O

Table 2: State-of-the-art in SSMA.

While there are still some further publications on the SSMA in the JSSP, to the best of the author's knowledge there are only the nine papers presented which were published on the SSMA in the JSSP. Concerning the works that include a consideration of dynamics, only two works can be found. In the MAS presented by Sahin et al. [64], which is based on pure self-organization through communication and negotiation between the agents, dynamics are considered, but the achieved results regarding the optimization of throughput time are considerably worse than with other approaches.

Methodology

Both centralized and decentralized approaches to production control offer different advantages and disadvantages. The decentralized approach to self-organization between different resources offers very high flexibility in stochastic environments and in responding to unexpected events but unlike a centralized approach, it does not have an overview at the entire (re-)manufacturing system, which reduces the likelihood of achieving global optimization of the RS. With a centralized control architecture, it is more likely to achieve global optimization, but on the other hand, this approach does not has the flexibility of a decentralized approach. In order to combine the advantages of both approaches, to do justice to the stochastic environment of a remanufacturing system and nevertheless to achieve a global optimization of the system, a hybrid architecture based on multi-agent technology for production control in FRS is presented in the following. The presented MAS contains basic elements of the PROSA reference architecture and integrates components of the approaches presented in [69], [70]. These approaches are modified in places and extended by the introduction of a *Transport Agent* and an *Expert Agent*. The *Expert Agent* enables the integration of the workers expert knowledge into the control system, to meet the special requirements in the domain of remanufacturing. If, due to unknown product condition, a process can not be executed as plant, the worker can look at the problem and decide, based on his expert knowledge, what would be a executable alternative process and inform the control system about his alternative decision. Within the Scheduler Agent, a cross-agent type schedule is created, which is generated using SSMA. The architecture of the MAS for controlling remanufacturing is a hierarchical, decentralized architecture, which is shown in Fig 3. Decisions about the control of the remanufacturing system can be made at both levels.

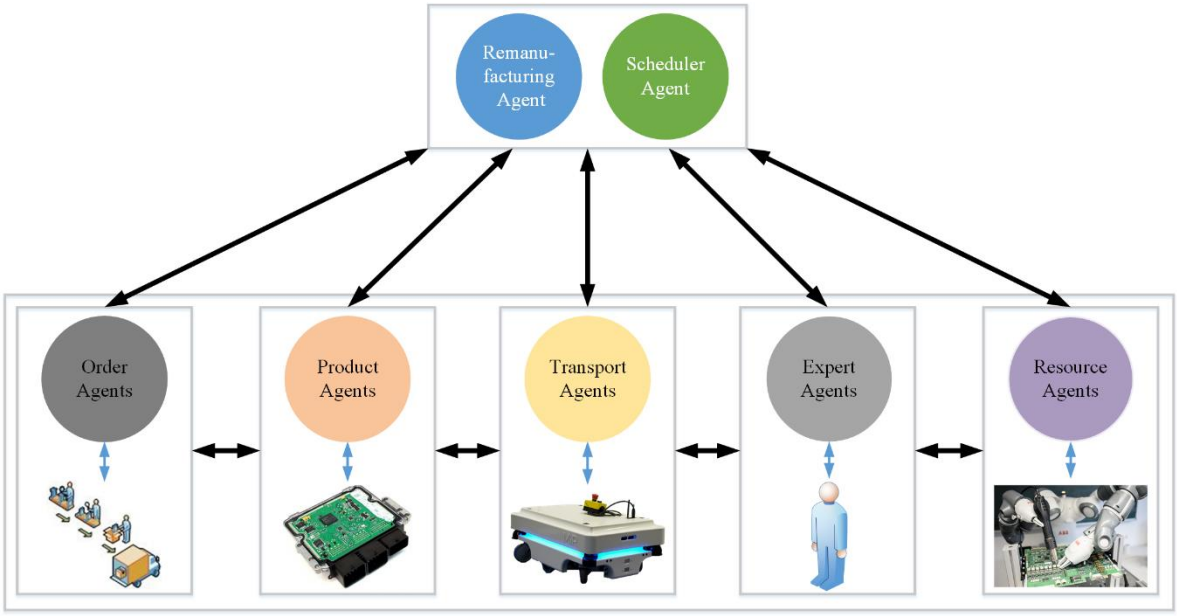


Fig 3: Architecture for the agent-based PPC in remanufacturing.

The proposed MAS consists of the agents and agent types listed below and shown in Fig 3:

- **Remanufacturing Agent:** Coordinates execution processes, scheduling and, if necessary, interactions between other agents;
- **Scheduler Agent:** Creates and optimizes simultaneous scheduling of manufacturing resources (machines, manual workstations) and AGVs.
- **Order Agents:** Manages separate orders within the remanufacturing system;
- **Product Agents:** Contains information about the product and its manufacturing capabilities;
- **Transport Agents:** Manages the flexible material handling systems (eg. AGVs)
- **Expert Agents:** Integration of the worker into the control system to solve problems when unexpected events occur through expert knowledge.
- **Resource Agents:** Manage the manufacturing resources (machines and manual workstations);

Constraint programming (CP) is used to carry out SSMA within the *Scheduler Agent*. This programming paradigm is a separate area of artificial intelligence and allows the modelling of the problem at hand through constraints and the integration of associated solution mechanisms. The big advantage of the CP is the simple adaptation of the problem to new boundary conditions. If complex metaheuristics are used, even small changes to the problem can lead to large, required changes.

The solving of SSMA by constraint programming combines on the one hand the scheduling of the machines in the form of a JSSP and on the other hand the associated organization of the necessary transports between the individual machines by a limited number of AGVs. The JSSP can be described by two constraints in the context of CP:

- **Precedence constraints:**
  This constraint results from the condition that, in the case of two consecutive operations within an job, the first must be completed before the second can be started.
- **No overlap constraint:**
  This constraint describes the condition that a machine can only process one operation at a given time.

The individual operations of the different jobs are represented by an *interval variable*. An interval variable is used to model a time interval in which a certain property is contained (an activity, eg. operation, is executed, a resource is unused by maintenance,...). The interval variable of an operation is described by the associated processing time. The individual transport orders are also represented by interval variables, whereby the corresponding time interval equals the transport time between the machine on which the next operation of the job is executed and the predecessor machine equals. The predecessor machine is the machine on which the previous operation of the job at hand was executed. If it is the first operation of a job, the time interval of the interval variable corresponds to the transport time between the warehouse and the machine on which the first operation of the job is executed. In the case of the last operation of an job, the time interval corresponds to the transport time between the machine processing the last operation of the job and the warehouse to which the job is transported after completion. The interval variables of the transfer orders created in this way are described through the following constraints:

- **Precedence constraints:**
    - An operation cannot be performed until the corresponding transport order has been completed;
    - A transport order cannot be executed until the predecessor machine has completed the corresponding operation.

In addition, the empty trips must be modelled and taken into account as well. Empty trips are trips during which an AGV is not transporting a product. If, for example, an AGV is at position A after it has completed a loaded trip there, and must carry out a material transport from B to C as the next order, than the trip from A to B is an empty trip. The time for the necessary empty trips must also be taken into account in the scheduling between two loaded trips of an AGV, since the second loaded trip can only be executed after the empty trip between the previous loaded trip and the current one has been completed. For a better understanding the following example should be considered. If transport order X is a transport (loaded trip) from C to A and the subsequent transport order Y is a transport from B to C, the AGV can only process order Y after the empty trip from A to B has been performed. These empty trips are modelled by a *transition distance matrix*. This is an *n* x *n* matrix, where *n* corresponds to the number of transport orders necessary for the realization of the existing orders. In this matrix, all possible empty trips between the individual transport orders are plotted. These empty trips are then taken into account as *transition distances* within the *no overlap constraint* of the individual AGVs integrated. This constraint describes the fact that transport orders of a AGV can not overlap and that the corresponding transition distance, if any, must be present between two consecutive transport orders. In addition, each transport order is assigned to all available AGVs and an *alternative constraint* describes that each transport order can be executed by any available AGV, but just one of them will be finally assigned to that specific transport order.

The constraint program described above is implemented in Java using the IBM ILOG CPLEX Optimization Studio 12.8.0 API. The Solver CP Optimizer is used to solve the problem. The metaheuristic algorithm *Large Neighbourhood Search (LNS)* is used to optimize the simultaneous scheduling. This procedure is an iterative improvement algorithm. This procedure starts the improvement of the solution based on a solution generated by simple design procedures. Within the individual iterations, individual fragments of the solution (operations) within a defined neighbourhood are interchanged to create a new solution. The selection of the fragments to be swapped is usually stochastic, so that different fragments of the solution are swapped at each iteration step. This can destroy the schedule by violating sequence or overlap conditions. For this reason, after a new solution has been created, it is repaired. The solution is changed until all boundary conditions are met again. The neighbourhood $N(x)$ of a solution x is defined as the set of solutions which can be reached by using the swapping of fragments and the subsequent repair of the new solution. The basic idea of the LNS is that the large neighbourhood makes it possible to move easily in the solution space, even if the problem is strongly limited by boundary conditions. Approaches based on small neighbourhoods, on the other hand, have much more difficulties in searching the solution space [168].

Results

The verification of the advantages resulting from SSMA compared to the sequential scheduling of these resources is carried out through simulation using the benchmark instances developed by Bilge and Ulusoy [53]. The benchmark instances used consist of four layout variants, which differ with regard to the arrangement of the individual machines and the associated travel times between the machines (see Fig 5**Fehler! Verweisquelle konnte nicht gefunden werden.** respectively Table 3) as well as ten job sets, which can be seen in Fig 4. Each of the ten job sets consist of four to eight jobs and 13 to 21 associated operations, which are executed on four machines and transported by two AGVs. Combining the four layouts with the ten job sets results in 40 different test instances. The job sets shown in Fig 4 are to be read in such a way that the value in brackets after the respective machine, which is represented by M1 to M4, indicates the process time of the corresponding operation on this machine. The number of transport jobs corresponds to the number of operations, since each job is transported in the first step from the L/U station (load/unload) to the machine that performs the first operation of the job and then between the machine processing the current operation and the machine processing the successor operation. These benchmark instances all have a *t/p* ratio (transport time/process time) of $t/p > 0.25$. This ratio has a great influence on the extent to which the material transport influences the scheduling with regard to the machines [53]. To consider a further t/p ratio, a new set of instances is generated by halving the travel times and multiplying the process times by a factor of two or three. A $t/p < 0.25$ results for all 42 benchmark instances obtained in this way. A total of 82 instances are thus available. An additional listing of the processing times of the jobs obtained in this way or of the transport times for the four layout variants is not provided here for the sake of clarity.

Job Set 1
Job 1: M1(8); M2(16); M4(12)
Job 2: M1(20); M3(10); M2(18)
Job 3: M3(12); M4(8); M1(15)
Job 4: M4(14); M2(18)
Job 5: M3(10); M1(15)

Job Set 2
Job 1: M1(10); M4(18)
Job 2: M2(10); M4(18)
Job 3: M1(10); M3(20)
Job 4: M2(10); M3(15); M4(12)
Job 5: M1(10); M2(15); M4(12)
Job 6: M1(10); M2(15); M4(12)

Job Set 3
Job 1: M1(16); M3(15)
Job 2: M2(18); M4(15)
Job 3: M1(20); M2(10)
Job 4: M3(15); M4(10)
Job 5: M1(8); M2(10); M3(15); M4(17)
Job 6: M2(10); M3(15); M4(8); M1(15)

Job Set 4
Job 1: M4(11); M1(10); M2(7)
Job 2: M3(12); M2(10); M4(8)
Job 3: M2(7); M3(10);M1(9); M3(8)
Job 4: M2(7); M4(8); M1(12); M2(6)
Job 5: M1(9); M2(7); M4(8); M2(10); M3(8)

Job Set 5
Job 1: M1(6); M2(12); M4(9)
Job 2: M1(18); M3(6); M2(15)
Job 3: M3(9); M4(3);M1(12)
Job 4: M4(6); M2(15)
Job 5: M3(3); M1(9)

Job Set 6
Job 1: M1(9); M2(11); M4(7)
Job 2: M1(19); M2(20); M4(13)
Job 3: M2(14); M3(20); M4(9)
Job 4: M2(14); M3(20); M4(9)
Job 5: M1(11); M3(16); M4(8)
Job 6: M1(10); M3(12); M4(10)

Job Set 7
Job 1: M1(6); M4(6)
Job 2: M2(11); M4(9)
Job 3: M2(9); M4(7)
Job 4: M3(16); M4(7)
Job 5: M1(9); M3(18)
Job 6: M2(13); M3(19); M4(6)
Job 7: M1(10); M2(9); M3(13)
Job 8: M1(11); M2(9); M4(8)

Job Set 8
Job 1: M2(12); M3(21); M4(11)
Job 2: M2(12); M3(21); M4(11)
Job 3: M2(12); M3(21); M4(11)
Job 4: M2(12); M3(21); M4(11)
Job 5: M1(10); M2(14), M3(18); M4(9)
Job 6: M1(10); M2(14), M3(18); M4(9)

Job Set 9
Job 1: M3(9); M1(12); M2(9); M4(6)
Job 2: M3(16); M2(11); M4(9)
Job 3: M1(21); M2(18); M4(7)
Job 4: M2(20); M3(22); M4(11)
Job 5: M3(14); M1(16), M2(13); M4(9)

Job Set 10
Job 1: M1(11); M3(19); M2(16); M4(13)
Job 2: M2(21); M3(16); M4(14)
Job 3: M3(8); M2(10); M1(14); M4(9)
Job 4: M2(13); M3(20); M4(10)
Job 5: M1(9); M3(16), M4(18)
Job 6: M2(19); M1(21), M3(11); M4(15)

Fig 4: Data concerning the job sets used as test problems.

Each of the four layout variants has a different arrangement of the four machines and the L/U station, resulting in different travel distances and therefor times between the individual machines (see Fig 5). M1 to M4 correspond to the respective processing machines and L/U (Load/Unload) represents the station where all jobs have to be picked up before their first processing. This is comparable to an goods-reception store.

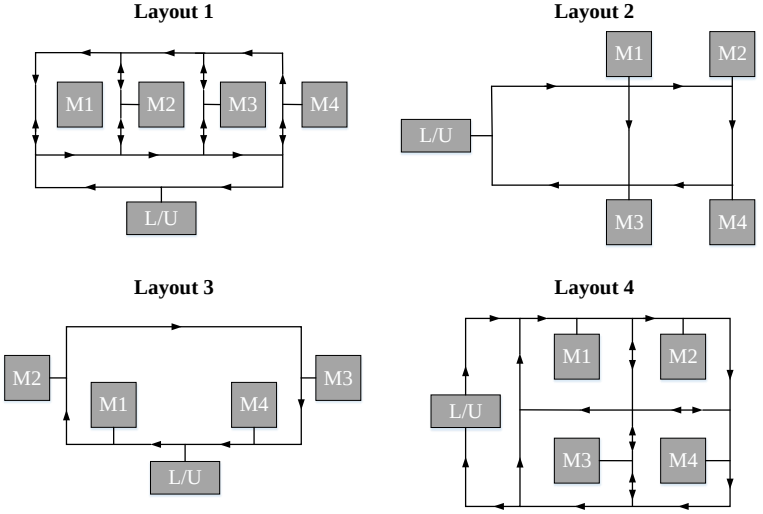


Fig 5: Different layouts within the benchmark from Bilge and Ulusoy [53].

The travel times between the individual machines are represented in the form of a *transport time matrix* (see Table 3). This transport time matrix is to be read in such a way that the transport time from a machine in the left column to a machine in the upper row corresponds to the value at which the respective row and column crosses. The transport times are unit less and given in the benchmark from [53].

	L/U	M1	M2	M3	M4
Layout 1					
L/U	0	6	8	10	12
M1	12	0	6	8	10
M2	10	6	0	6	8
M3	8	8	6	0	6
M4	6	10	8	6	0
Layout 2					
L/U	0	4	6	8	6
M1	6	0	2	4	2
M2	8	12	0	2	4
M3	6	10	12	0	2
M4	4	8	10	12	0
Layout 3					
L/U	0	2	4	10	12
M1	12	0	2	8	10
M2	10	12	0	6	8
M3	4	6	8	0	2
M4	2	4	6	12	0
Layout 4					
L/U	0	4	8	10	14
M1	18	0	4	6	10
M2	20	14	0	8	6
M3	12	8	6	0	6
M4	14	14	12	6	0

Table 3: Transport time matrix for the test problem according to Bilge and Ulusoy [53].

The problem, based on these benchmark, can be described as follows. There are a number of machines as well as AGVs and a number of jobs to be processed. Each job has a number of precedence constraints and each machine can process at most one operation at a time. It is assumed that the setup times are included in the process times and that operation cannot be interrupted after the start of processing.
The following assumptions are taken into account in this study.

- The fleet size is two AGVs.
- Processing, set-up and loading times are deterministic.
- All AGVs are identical in terms of speed and performance.
- AGVs transport only one product per trip.
- Problems such as traffic control, congestion, machine failure or downtimes, rejects, rework and vehicle disposition for battery charging are not taken into account here.
- The loading and unloading of jobs on the machine is only carried out by the AGVs.
- The AGVs are reliable and free from any malfunctions in operation.

The optimization criterion, respectively the target function, of the used benchmark is the makespan, which is the timespan to complete all jobs within a certain job set.
This simultaneous scheduling approach of machines and AGVs is compared with the sequential scheduling, currently used in the industry, to assess possible advantages with regard to a reduction in makespan as a result. In the sequential scheduling approach, the first step is to optimize the schedule regarding the jobs and their operations to be processed on the assigned machines. Subsequently, a fleet management system assigns the individual available AGVs to the transport orders resulting from the schedule regarding the machines like shown in Fig 6.

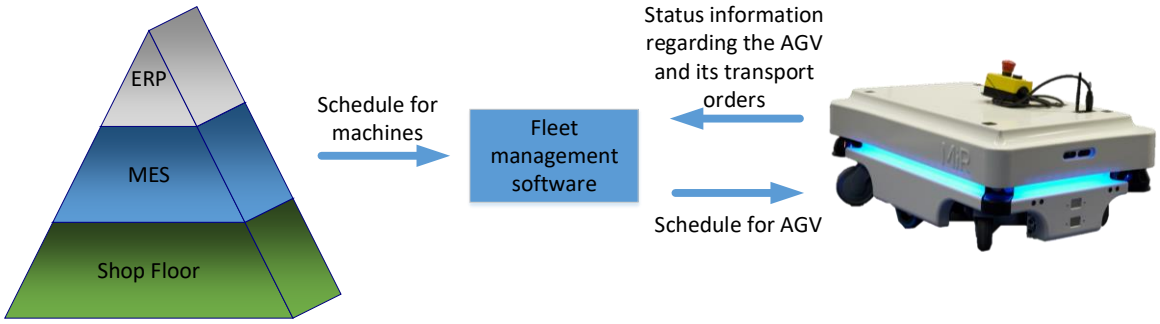


Fig 6: Sequential scheduling using MES and Fleet Management Software.

The benchmark instances presented above are used for this purpose. In order to ensure the comparability of the results, the simultaneous and sequential scheduling of machines and AGVs is implemented as a constraint program in Java and solved with the Solver CP Optimizer from IBM. The constraint programs described in previous chapter, for modelling the JSSP and SSMA, are implemented.
By using CP Optimizer it is possible to find the optimum for the JSSP and the SSMA, including verifications. In simultaneous scheduling, the JSSP and the assignment of transport orders are considered together and optimized with regard to minimizing the makespan, including the consideration of transport times. Sequential scheduling involves first optimizing the makespan for the JSSP and then, based on the best solution found for the JSSP, assigning the AGV to the transport orders. The assignment of the AGVs to the individual transport orders, normally done by the fleet management software, is done by the *Vehicle Assignment Heuristic (VAH)* presented in [63]. The VAH proceeds as follows when assigning the available AGVs:

1. Identification of the position (current AGV location) and the time at which the AGV is available again (VRT).
2. Calculation of the travel time from the current position of the AGV to the machine on which the job (the product) is currently located.
3. Addition of the travel time to the VRT to calculate the completion time of the empty trip (VET).
4. Check whether the current operation of the job is completed or not; if necessary, the AGV waits until the operation is completed.
5. Compare the completion time of the operation and the VET, for further calculations the higher of these two times will be used.
6. Calculation of the travel time between the machine on which the job is currently located and the machine on which the next operation of the job will be executed.
7. Add this travel time to the value obtained in step 5 resulting in the end time of the loaded trip (VLT).
8. As soon as the loaded trip is finished, the vehicle is ready for its next assignment and the VLT of the current trip becomes the VRT for the next trip. The heuristic selects the vehicle with the lowest VLT value.

The VAH is implemented in Java. The JSSP solution found by the CP Optimizer Solver is imported into Java via the Java API of IBM ILOG CPLEX Optimization Studio 12.8.0 in order to assign the AGVs to the resulting transport orders through the VAH.
In addition to the simultaneous and sequential scheduling of machines and AGVs, the approach of self-organization using MAS, which was presented in [55], was also included in the comparison. The results for the benchmark instances regarding self-organization were extracted from the publication of Erol et al. [55].
The average results can be found in Fig 7 and the exact results, concerning the individual test instances, can be found in Table 13 (Appendix). The comparison shows that the SSMA results in a reduction of the average makespan of 19.7 %. The comparison between sequential scheduling and self-organization using a MAS shows an increase of 10.4 % regarding the makespan.

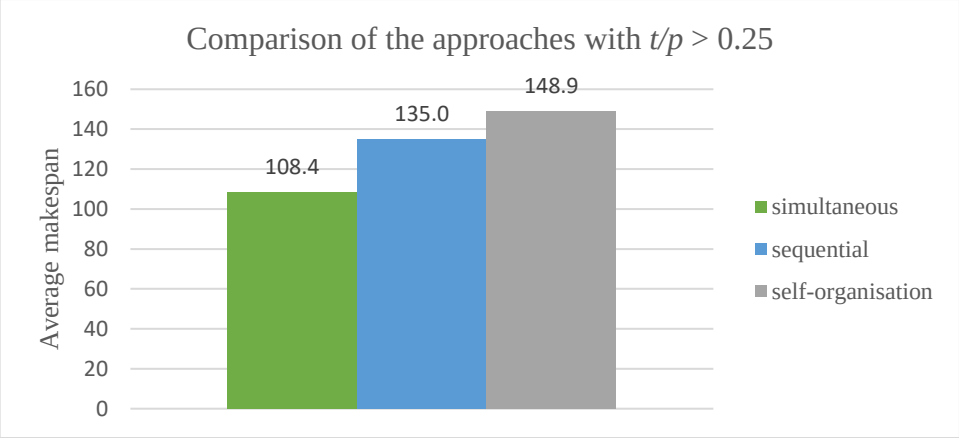


Fig 7: Results of the comparison between simultaneous, sequential and self-organizing [55] *scheduling of machines and AGVs for benchmark instances with a t/p ratio > 0.25.*

Related to the results for the instances with a t/p ratio < 0.25, it can be seen that the optimization potential is lower here due to simultaneous scheduling. This result in an average reduction of the makespan due to simultaneous scheduling of 7.1 % compared to sequential scheduling. The use of a self-organizing system, on the other hand, results in a 4.9 % worse result compared to sequential scheduling.

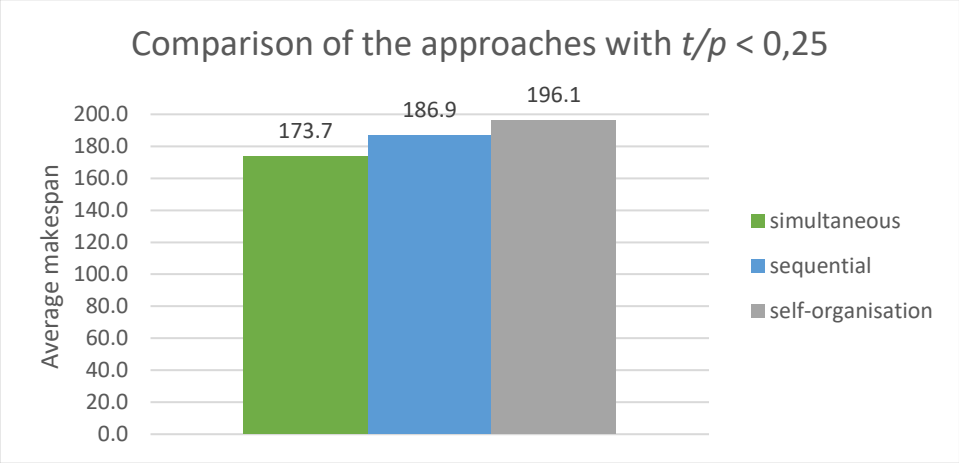


Fig 8: Results of the comparison between simultaneous, sequential and self-organizing [71] scheduling of machines and AGVs for benchmark instances with a t/p ratio < 0.25.

The optimization potential regarding the reduction of the makespan through SSMA is significantly higher with a t/p > 0.25 than with a t/p < 0.25. This is because the transport time with a t/p < 0.25 has a lower share of the total makespan than with a t/p > 0.25, whereby the scheduling of the transport tasks also has a low share of the overall performance of the system. Nevertheless, a reduction of the makespan can be achieved by using simultaneous scheduling. In both t/p ratios examined, no significant improvement can be achieved by self-organization using a MAS. In relation to the instances with a t/p < 0.25, there is even a deterioration compared to sequential scheduling. The reason for this is that the omission of a central unit in this application means that there is no overall overview of the system, making global optimization unlikely. This has a particular effect on instances with t/p < 0.25, since the scheduling for the machines was optimally planned here and the subsequent transport orders assigned by the VHA have only a minor influence on the overall system due to the low t/p ratio.

In the previous study it can be seen that the optimization potential regarding the makespan reduction through simultaneous scheduling, in comparison to sequential scheduling, is higher for the benchmark instances with a t/p > 0.25 than for the instances with a t/p < 0.25. To further investigate if and to what extent there is a relationship between the t/p ratio and the percentage optimization potential for makespan reduction by means of simultaneous scheduling, the achieved makespan reduction over the corresponding t/p ratio for all benchmark instances is shown in Fig 9 (t/p < 0.25) and Fig 10 (t/p < 0.25). Each blue dot represents one of the 82 benchmark instances with the corresponding perecentage makespan reduction (ordinate) and the corresponding t/p ratio (abscissa). The dashed line to be seen in each case is the trend line, which is calculated using a simple linear regression. This shows in both tables that an increase in the t/p ratio is accompanied by an increase in the reduction of the makespan through SSMA.

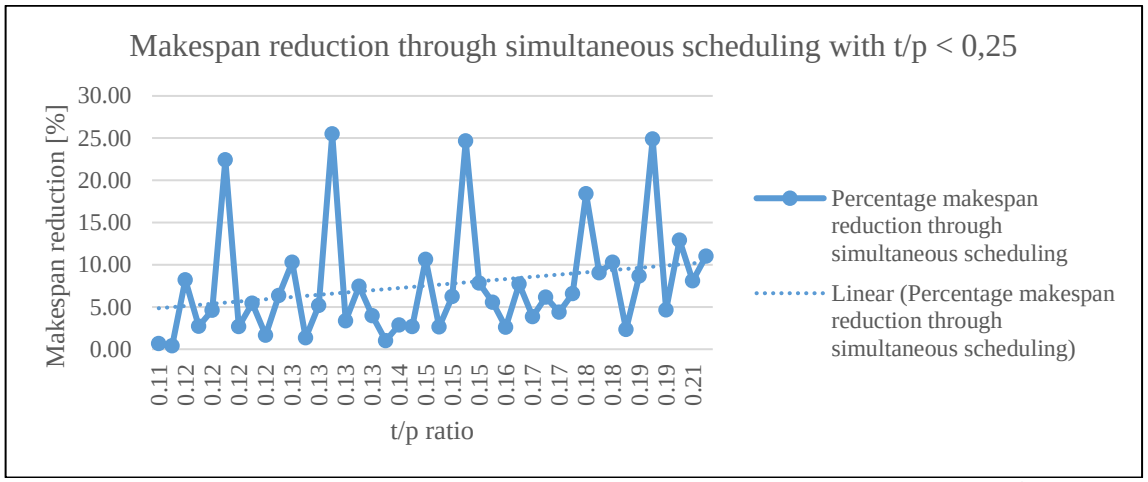


Fig 9: Percentage makespan reduction through simultaneous scheduling in comparison to sequential scheduling for benchmark instances with t/p < 0.25.

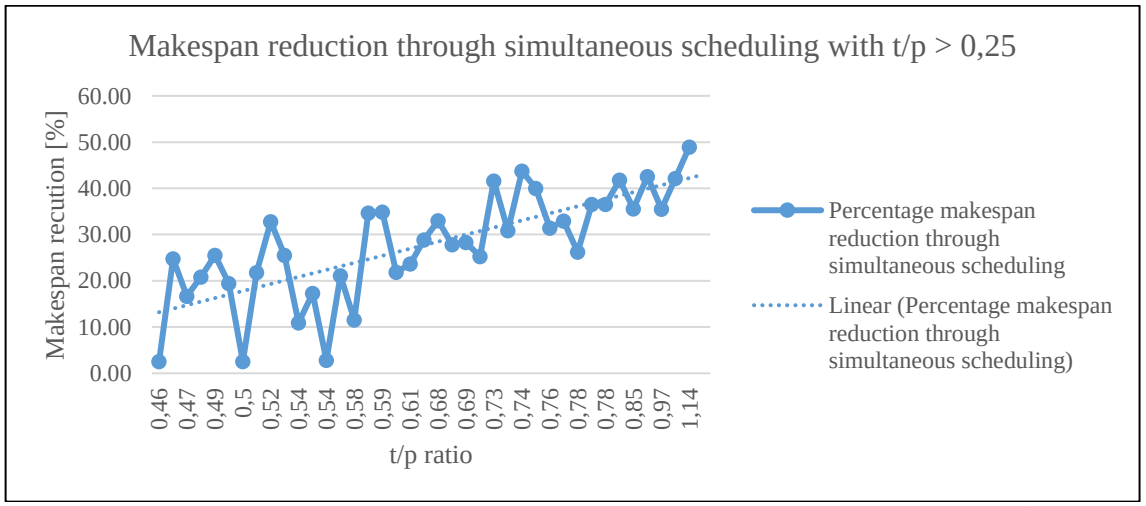


Fig 10: Percentage makespan reduction through simultaneous scheduling in comparison to sequential scheduling for benchmark instances with t/p > 0.25.

Looking at real manufacturing systems, this correlation shows that even higher optimization potentials are realistic than in the benchmark instances used. The reason for this is that the t/p ratio in real manufacturing systems is higher than the t/p ratio of the instances used. The makespan of a real job shop production is composed of the following time components [71]:

- Idle/waiting time: 70%.
- Transport time: 20%.
- Processing time: 10%.

This corresponds to a t/p ratio of 2.0.

After examine the potential of makespan reduction through SSMA, compared to the currently used sequential scheduling, within the next part the CP approach presented is compared to other known works and methods for solving this simultaneous scheduling problem.

Regarding the scheduling, especially in the context of production control, not only the quality of the solution found is decisive, but also various other criteria. One important criterion here is the computational time of the used optimization algorithm. This indicates how long the algorithm takes to find a solution. Especially in real manufacturing systems, in which unpredicted events such as machine failures, other malfunctions or planning changes frequently occur, it is essential that the optimization algorithm used for scheduling can deliver a good, new solution within an appropriate time. For this reason, in this simulation study, not only the makespan is used as an evaluation criterion, but also the computational time (referred to as CPU time) for comparing the different optimization algorithms is considered. However, the computational times are not given in all works considered for comparison. The benchmark instances from [149] already presented serve also as a basis for this comparison.

The presented approach of constraint programming was implemented in Java and solved using the Solvers *CP Optimizer* from *IBM*. The hardware used is an *Intel® Core™ i5-7200U CPU* with 2.50 GHz and 8.00 GB RAM. For comparison, the hardware used by Chaudhy and Shami [153] and Huang [170] is also listed below:

- Chaudhy [153]: *PIV* CPU with 1.7 GHz and 128 MB RAM
- Huang [170]: *Intel Xeon* CPU with 2.40 GHz and 16 GB RAM

The different test instances with t/p > 0.25 are described by "EX" followed by two numbers (see Table 4). The first number represents the job set and the second number the layout used. The test instances with a t/p < 0.25 (see Table 5) are also designated by "EX", but are now followed by three numbers. Analogous to the previous scheme, the first number represents the order set used, the second number the layout used and the third number added represents the factor by which the process times were multiplied. This number is 0 if the process times have been multiplied by two and 1 if the process times have been multiplied by three. This creates 42 additional test instances, making a total of 82 test instances. The listed results, the methods used for comparison, are taken from the respective publications and all refer to the application of the respectively presented method to the benchmark instances of Bilge and Ulusoy [53]. The method used for comparison is Erol et al. [55] which proposes an approach based on multi-agents, whereby a pure self-organization was realized. Lin et al. [65] use an approach of simulation-based optimization (SBO). (Meta-)heuristic methods are used by Zheng et al. [72] (tabu search) and by Chaudhry et al. [73] (genetic algorithm). Huang [74] uses MILP to solve the scheduling problem as a representative of the exact procedures. In the investigations of [65], [55] and [72] there is no consideration of computational time. In order to verify the performance of the presented constraint programming method, the best solution found within 1 sec computational time is displayed (CPU < 1s) for instances where our algorithm took longer than 1 sec to find the presented solution.

	Erol	**Lin**	**Zheng**	**Chaudry**		**Huang**		**Groß**		
	MAS	**SBO**	**TS**	**GA**		**MILP**		**CP**		
Instance	MS	MS	MS	MS	CPU	MS	CPU	MS	CPU	CPU < 1s
EX11	130	96	96	96	7	96	30,58	96	0,05	
EX21	143	100	100	100	113	100	730,77	100	0,41	
EX31	142	100	99	99	34	99	176,83	99	0,85	
EX41	198	114	112	112	193	112	50803,3	112	0,97	
EX51	130	87	87	87	68	87	136,43	87	0,05	
EX61	153	118	118	118	1260	118	7927,26	118	0,51	
EX71	129	111	111	115	104	-	-	111	529,3	127
EX81	196	161	161	161	13	161	27,79	161	0,03	
EX91	178	116	116	116	41	116	22,09	116	0,02	
EX101	188	153	146	150	75	146	7138,1	146	0,48	
EX12	98	82	82	82	6	82	4,34	82	0,02	
EX22	86	76	76	76	18	76	5,44	76	0,02	
EX32	114	85	85	85	15	85	8,3	85	0,02	
EX42	129	86	87	88	54	87	3118,96	87	0,34	
EX52	98	69	69	69	10	69	17,82	69	0,19	
EX62	123	98	98	98	120	98	10,18	98	0,91	
EX72	92	79	79	81	240	79	11915	79	3,58	84
EX82	172	151	151	151	4	151	14,77	151	0,02	
EX92	123	102	102	102	20	102	9,69	102	0,23	
EX102	154	135	135	141	300	135	161,63	135	0,03	
EX13	109	84	84	84	18	84	8,14	84	0,03	
EX23	98	86	86	86	7	86	95,98	86	0,04	
EX33	103	86	86	86	54	86	6,68	86	0,03	
EX43	155	89	89	89	25	89	3997,25	89	0,03	
EX53	109	74	74	74	45	74	83,23	74	0,04	
EX63	128	103	103	104	1200	103	23,33	103	0,03	
EX73	93	82	83	90	300	83	33725,1	83	6,38	89

EX83	172	153	153	153	5	153	14,45	153	0,03	
EX93	119	105	105	105	21	105	10,17	105	0,37	
EX103	158	139	137	140	83	137	290,78	137	0,74	
EX14	168	103	103	103	23	103	27,67	103	0,84	
EX24	169	108	108	108	11	108	3698,61	108	0,66	
EX34	167	110	111	111	18	111	832,66	111	35,5	112
EX44	242	126	121	126	68	121	22554,1	121	0,82	
EX54	168	96	96	96	29	96	176,06	96	2,3	98
EX64	189	120	120	122	12	120	1760,19	120	2,17	122
EX74	156	126	126	130	32	-	-	127	2,36	133
EX84	251	163	163	163	38	163	4681,18	163	0,08	
EX94	181	122	120	120	42	120	61,69	120	0,19	
EX104	246	159	157	159	161	157	79885	157	2,23	169
Avg.	**148,9**	**108,8**	**108,4**	**109,4**	**122,2**	**107,8**	**6162,9**	**108,4**	**14,8**	
Median					**36**		**116,2**		**0,29**	

Table 4: Comparison of the results of various optimization methods for SSMA with t/p > 0.25. MS = makespan; CPU = computational time

In the results of [74] it should be noted that this procedure has not found a solution for all instances, which is indicated by a "-" in the corresponding line. This circumstance also results in a better average makespan compared to other methods, because if only the results for the individual instances are compared, it can be stated that no better result could be found for any instance. Apart from the fact that the exact procedure does not find a solution for all instances, the computational times for finding a solution are very high. For example, instance EX41 requires more than 14 hours finding a solution. The advantage of this method is that the solutions found are the optimal solution to the problem at hand. The results presented by Erol [55] are significantly behind those of the other approaches. The approaches of Lin [65], Zheng [72] and Chaudhry et al. [73] each provide similarly good results. The computational times of [73], with an average value of 122.2s and a median of 36s, are clearly below the values of [74] with an average computational time of 6162.9s and a median of 116.2s for the instances, respectively.

	Erol	Chaudry	Groß		
	MAS	GA	CP		
Instance	MS	MS	MS	CPU	CPU < 1s
EX110	135	126	126	0,02	
EX210	157	148	148	0,02	
EX310	154	150	150	0,07	
EX410	211	119	119	0,05	
EX510	118	102	102	0,02	
EX610	204	186	186	0,12	
EX710	138	137	137	0,07	
EX810	330	292	292	0,03	
EX910	191	176	176	0,08	
EX1010	269	238	238	0,6	
EX120	127	123	123	0,02	
EX220	151	143	143	0,02	
EX320	144	145	145	0,09	
EX420	161	114	114	0,07	
EX520	110	100	100	0,01	
EX620	196	181	181	0,03	
EX720	132	136	136	0,02	

EX820	319	287	287	0,03	
EX920	187	173	173	0,04	
EX1020	266	236	236	15,92	237
EX130	134	122	122	0,02	
EX230	151	146	146	0,02	
EX330	129	146	146	0,01	
EX430	228	114	114	0,02	
EX530	111	99	99	0,01	
EX630	198	182	182	0,08	
EX730	132	137	137	0,03	
EX830	273	288	288	0,03	
EX930	187	174	174	0,04	
EX1030	266	237	237	1,03	238
EX140	137	124	124	0,02	
EX241	230	217	217	0,02	
EX340	155	151	151	0,04	
EX341	227	221	221	0,05	
EX441	344	172	172	0,04	
EX541	158	148	148	0,03	
EX640	211	184	184	0,23	
EX740	158	137	137	0,02	
EX741	206	203	203	0,03	
EX840	331	293	293	0,02	
EX940	195	175	175	0,06	
EX1040	276	240	240	32,06	241

Avg.	**196,1**	**173,7**	**173,7**	**1,3**	
Median				**0,03**	

Table 5: Comparison of the results of various optimization methods for SSMA with t/p < 0.25. MS = makespan; CPU = computational time

The approach presented in this paper provides the best results in terms of both makespan and computational time. With an average makespan of 108.4, it provides the same results as Zheng [72]. Relative to computational time, the approach [73] and [74] with an average makespan of 14.8s and a median of 0.29s is clearly superior. Regarding the instances with t/p < 0.25, the presented approach achieves an average makespan of 1.3s, as well as a median of 0.03s. For all instances where the computaional time was more than one second, it can be seen that the approach is able to find solutions close to the optimum within one second.

Conclusions

The paper at hand presents an agent-based hybrid control architecture for the special demands in the domain of remanufacturing. A centralized scheduling algorithm was proposed which also takes the scheduling of AGVs into consideration to enable the often-required flexible material handling within a remanufacturing system. In contrast to currently available control and fleet management software the scheduling of machines and AGVs is not executed sequential, but simultaneously. Because these both scheduling problems depend and influence each other, the simultaneous scheduling approach generates better results regarding the makespan optimization then the sequential approach. Simulation studies using benchmark instances show that the simultaneous scheduling approach generates on average results with a 19.7 % reduced makespan compared to sequential scheduling. Furthermore the simultaneous approach was compared to self-organisation using MAS with the result of an on average 27.2 % lower makespan for the simultaneous approach. This also shows why the use of a hybrid control architecture is advantageous, as significantly better scheduling plans can be achieved compared to the pure, decentralized self-organization. It was also shown, using the same benchmark instances, that the higher the t/p ratio the higher the percentage makespan reduction

through simultaneous scheduling. With benchmark instances varying from a t/p ration between 0,11 and 1,14 and an percentage makespan reduction between 1 % and 49 % and the fact that real manufacturing systems have a t/p ration of around 2,00 it can be assumed that the optimization potential in real remanufacturing systems is even higher than for the used benchmark instances.

Furthermore, the proposed optimization approach for the SSMA using Constraint Programming was compared to other state-of-the-art works in the field of simultaneous scheduling. It could be shown that our approach produces the schedules with the best makespan and this within the shortest computational time of the compared approaches.

In further work, the simultaneous scheduling approach will also be test with real data from an remanufacturing plant to verify the superiority of simultaneous scheduling in a real world scenario. In addition, the behavior of the approach will be investigated when unexpected events such as machine failures and new orders occur. A new schedule adapted to the changed boundary conditions will be created. In addition to the investigations by simulation studies, the approach will be also implemented in a demonstrator. This demonstrator has already been implemented in part, among other things the control of an AGV by the simultaneously generated schedule.

Acknowledgements

This work was funded by „Europäischer Fond für regionale Entwicklung“ (EFRE) in the scope of the INTERREG V A Program Greater Region within the Project “Robotix-Academy”.

References

[1] Shiyong Wang, Jiafu Wan, Di Li, and Chunhua Zhang, “Implementing Smart Factory of Industrie 4.0: An Outlook,” *Int. J. Distrib. Sens. Networks*, vol. 2016, 2016.

[2] A. G. Schuh *et al.*, “Cyber Physical Production Control,” in *Industrial Internet of Things: Cybermanufacturing Systems*, S. Jeschke, C. Brecher, H. Song, and D. B. Rawat, Eds. Cham: Springer International Publishing, 2017, pp. 519–539.

[3] W. Dangelmeier, U. Pape, and M. Rüther, “Agentensysteme für das Supply Chain Management,” 2004.

[4] A. Tharumarajah, A. J. Wells, and L. Nemes, “Comparison of emerging manufacturing concepts,” 1998, vol. 1, pp. 325–331 vol.1.

[5] P. Vrba and V. Marík, “Capabilities of Dynamic Reconfiguration of Multiagent-Based Industrial Control Systems,” *Syst. Man Cybern. Part A Syst. Humans, IEEE Trans.*, vol. 40, pp. 213–223, 2010.

[6] P. Leitão, “Agent-Based Distributed Manufacturing Control: A State-of-the-Art Survey,” *Eng. Appl. Artif. Intell.*, vol. 22, pp. 979–991, 2009.

[7] P. Lou, S. K. Ong, and A. Nee, “Agent-based distributed scheduling for virtual job shops,” *Int. J. Prod. Res. - INT J PROD RES*, vol. 48, pp. 3889–3910, 2010.

[8] A. Kouider and B. Bouzouia, “Multi-agent job shop scheduling system based on co-operative approach of idle time minimisation,” *Int. J. Prod. Res. - INT J PROD RES*, vol. 50, pp. 409–424, 2012.

[9] S. Wang, J. Wan, D. Zhang, D. Li, and C. Zhang, “Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination,” *Comput. Networks*, vol. 101, pp. 158–168, 2016.

[10] M. E. Leusin, M. Kück, E. M. Frazzon, M. U. Maldonado, and M. Freitag, “Potential of a Multi-Agent System Approach for Production Control in Smart Factories,” *IFAC-PapersOnLine*, vol. 51, no. 11, pp. 1459–1464, 2018.

[11] L. Sterling and K. Taveter, “The Art of Agent-Oriented Modeling,” 2009.

[12] J. Zhang, “Multi-Agent-Based Production Planning and Control,” in *Multi-Agent-Based Production Planning and Control*, 2017.

[13] M. Wooldridge, “An Introduction to MultiAgent Systems,” 2002, p. 348.

[14] K. Z. Gao, P. N. Suganthan, T. J. Chua, C. S. Chong, T. X. Cai, and Q. K. Pan, “A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion,” *Expert Syst. Appl.*, vol. 42, no. 21, pp. 7652–7663, 2015.

[15] J. Gao, L. Sun, and M. Gen, “A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems,” *Comput. Oper. Res.*, vol. 35, no. 9, pp. 2892–2907, 2008.

[16] S. V. Kamble, S. U. Mane, and A. J. Umbarkar, “Hybrid Multi-Objective Particle Swarm Optimization for Flexible Job Shop Scheduling Problem,” *Int. J. Intell. Syst. Appl.*, vol. 7, no. 4, pp. 54–61, 2015.

[17] J. Q. Li, Q. K. Pan, and K. Z. Gao, “Pareto-based discrete artificial bee colony algorithm for multi-

objective flexible job shop scheduling problems," *Int. J. Adv. Manuf. Technol.*, vol. 55, no. 9–12, pp. 1159–1169, 2011.
[18] L.-N. Xing, Y.-W. Chen, P. Wang, Q.-S. Zhao, and J. Xiong, "A Knowledge-Based Ant Colony Optimization for Flexible Job Shop Scheduling Problems," *Appl. Soft Comput.*, vol. 10, no. 3, pp. 888–896, 2010.
[19] M. Yazdani, M. Amiri, and M. Zandieh, "Flexible job-shop scheduling with parallel variable neighborhood search algorithm," *Expert Syst. Appl.*, vol. 37, no. 1, pp. 678–687, 2010.
[20] Q. Zhang, H. Manier, and M.-A. Manier, "A genetic algorithm with tabu search procedure for flexible job shop scheduling with transportation constraints and bounded processing times," *Comput. Oper. Res.*, vol. 39, no. 7, pp. 1713–1723, Jul. 2012.
[21] H. E. Nouri, O. Belkahla Driss, and K. Ghédira, "Controlling a Single Transport Robot in a Flexible Job Shop Environment by Hybrid Metaheuristics," *LNCS Trans. Comput. Collect. Intell.*, vol. 28, pp. 93–115, 2018.
[22] B. Ramasubramanian, "Re-entrant Manufacturing Systems," pp. 1–27.
[23] M. L. Junior and M. G. Filho, "Production planning and control for remanufacturing: literature review and analysis," *Prod. Plan. Control*, vol. 23, no. 6, pp. 419–435, 2012.
[24] C. Zikopoulos, "Remanufacturing lotsizing with stochastic lead-time resulting from stochastic quality of returns," *Int. J. Prod. Res.*, vol. 55, pp. 1–23, 2016.
[25] R. Zhang, S. K. Ong, and A. Y. C. Nee, "A simulation-based genetic algorithm approach for remanufacturing process planning and scheduling," *Appl. Soft Comput.*, vol. 37, pp. 521–532, Dec. 2015.
[26] P. Golinska-dawson, M. Kosacka, and A. Nowak, "Automotive Parts Remanufacturing – Experience of Polish Small Automotive Parts Remanufacturing – Experience of Polish Small Companies," no. January, 2015.
[27] V. D. R. Guide, "Production planning and control for remanufacturing: industry practice and research needs," *J. Oper. Manag.*, vol. 18, no. 4, pp. 467–483, 2000.
[28] M. Andrew-Munot, A. Yassin, S. T. Syed Shazali, M. Sawawi, S. J. Tanjong, and N. Razali, "Analysis of production planning activities in remanufacturing system," *J. Mech. Eng. Sci.*, vol. 12, no. 2, pp. 3548–3565, 2018.
[29] Y. Cui, Z. Guan, C. He, and L. Yue, "Research on remanufacturing scheduling problem based on critical chain management," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 215, no. 1, 2017.
[30] M. Lage Junior and M. Godinho Filho, "Master disassembly scheduling in a remanufacturing system with stochastic routings," *Cent. Eur. J. Oper. Res.*, vol. 25, no. 1, pp. 123–138, 2017.
[31] F. Ehm, "A data-driven modeling approach for integrated disassembly planning and scheduling," *J. Remanufacturing*, 2018.
[32] J. Jungbluth, W. Gerke, and P. Plapper, "Recent Progress Toward Intelligent Robot Assistants for Non-Destructive Recent Progress Toward Intelligent Robot Assistants for Non- Destructive Disassembly," *2. RACIR – Robot. Conf. Ind. Robot.*, no. June, 2018.
[33] P. He, "Optimization and Simulation of Remanufacturing Production Scheduling under Uncertainties," *Int. J. Simul. Model.*, vol. 17, no. 4, pp. 734–743, 2018.
[34] J. M. Kim, Y. D. Zhou, and D. H. Lee, "Priority scheduling to minimize the total tardiness for remanufacturing systems with flow-shop-type reprocessing lines," *Int. J. Adv. Manuf. Technol.*, vol. 91, no. 9–12, pp. 3697–3708, 2017.
[35] H. Wen, S. Hou, Z. Liu, and Y. Liu, "An optimization algorithm for integrated remanufacturing production planning and scheduling system," *Chaos, Solitons & Fractals*, vol. 105, pp. 69–76, 2017.
[36] S. Luo, G. Luo, and X. Zhao, "Common production process modeling for MES based on multi-Agent," *IEEE Int. Conf. Ind. Eng. Eng. Manag.*, pp. 1582–1586, 2014.
[37] M. Merdan, A. Zoitl, G. Koppensteiner, and M. Melik-Merkumians, "Adaptive Produktionssysteme durch den Einsatz von autonomen Softwareagenten," *Elektrotechnik und Informationstechnik*, vol. 129, no. 1, pp. 53–58, 2012.
[38] N. Vojdani, B. Erichsen, and T. Lück, "Using production real time data - An agent-based detailed planning by means of simulation," *Logist. J.*, pp. 1–8, 2017.
[39] R. M. Lima, R. M. Sousa, and P. J. Martins, "Distributed production planning and control agent-based system," *Int. J. Prod. Res.*, vol. 44, no. 18–19, pp. 3693–3709, 2006.
[40] B. Scholz-Reiter and M. Freitag, "Autonomous Processes in Assembly Systems," *CIRP Ann. - Manuf. Technol.*, vol. 56, no. 2, pp. 712–729, 2007.
[41] N. . He, D. Z. . Zhang, and Q. :. Li, "Agentbased hierarchical production planning and scheduling in make-to-order manufacturing system," *Int. J. Prod. Econ.*, vol. 149, pp. 117–130, 2014.
[42] Mobile Industrial Robots, "MiRFleet | Mobile Industrial Robots." [Online]. Available:

https://www.mobile-industrial-robots.com/de/products/mir-add-ons/mirfleet/. [Accessed: 25-Mar-2019].
[43] K. AG, “KUKA.NavigationSolution | KUKA AG.” [Online]. Available: https://www.kuka.com/en-gb/products/mobility/navigation-solution. [Accessed: 25-Mar-2019].
[44] BA Systèmes, “AGV Manager: AGV fleet management software | BA Systèmes.” [Online]. Available: https://www.basystemes.com/en/agvs/supervision/. [Accessed: 25-Mar-2019].
[45] DEMATIC, “E’tricc - Cutting Edge AGV Management Software.” [Online]. Available: http://egeminusa.com/automated-guided-vehicles/software/etricc/. [Accessed: 25-Mar-2019].
[46] Gebocermex, “AGV supervision system.” [Online]. Available: https://www.gebocermex.com/en/intralogistic-systems/automated-guided-vehicles/agv-supervision-system-pd-111. [Accessed: 25-Mar-2019].
[47] JBT, “SGV Manager - JBT.” [Online]. Available: https://www.jbtc.com/automated-systems/products-and-applications/products/software-and-services/software/sgv-manager. [Accessed: 25-Mar-2019].
[48] savant automation, “Controls & Software | Savant Automation.” [Online]. Available: http://www.agvsystems.com/controls-software/. [Accessed: 25-Mar-2019].
[49] S. C. Srivastava, A. K. Choudhary, S. Kumar, and M. K. Tiwari, “Development of an intelligent agent-based AGV controller for a flexible manufacturing system,” *Int. J. Adv. Manuf. Technol.*, vol. 36, no. 7–8, pp. 780–797, 2008.
[50] E. Cardarelli, V. Digani, L. Sabattini, C. Secchi, and C. Fantuzzi, “Cooperative cloud robotics architecture for the coordination of multi-AGV systems in industrial warehouses,” *Mechatronics*, vol. 45, pp. 1–13, 2017.
[51] F. Yao, A. Keller, M. Ahmad, B. Ahmad, R. Harrison, and A. W. Colombo, “Optimizing the Scheduling of Autonomous Guided Vehicle in a Manufacturing Process,” *Proc. - IEEE 16th Int. Conf. Ind. Informatics, INDIN 2018*, pp. 264–269, 2018.
[52] M. R. Garey, D. Johnson, and R. Sethi, “The Complexity of Flowshop and Jobshop Scheduling,” *Math. Oper. Res. - MOR*, vol. 1, pp. 117–129, 1976.
[53] Ü. Bilge and G. Ulusoy, “A Time Window Approach to Simultaneous Scheduling of Machines and Material Handling System in an FMS,” *Oper. Res.*, vol. 43, no. 6, pp. 1058–1070, 1995.
[54] M. Nageswararao, K. Narayanarao, and G. Ranagajanardhana, “Simultaneous Scheduling of Machines and AGVs in Flexible Manufacturing System with Minimization of Tardiness Criterion,” *Procedia Mater. Sci.*, vol. 5, pp. 1492–1501, 2014.
[55] R. Erol, C. Sahin, A. Baykasoglu, and V. Kaplanoglu, “A multi-agent based approach to dynamic scheduling of machines and automated guided vehicles in manufacturing systems,” *Appl. Soft Comput.*, vol. 12, no. 6, pp. 1720–1732, 2012.
[56] M. Mousavi, H. J. Yap, S. N. Musa, F. Tahriri, and S. Z. Md Dawal, “Multi-objective AGV scheduling in an FMS using a hybrid of genetic algorithm and particle swarm optimization,” *PLoS One*, vol. 12, no. 3, pp. 1–24, 2017.
[57] I. A. Chaudhry, “A Genetic Algorithm Approach for Process Planning and Scheduling in Job Shop Environment,” *Proc. World Congr. Eng.*, vol. III, pp. 2–6, 2012.
[58] D. B. M. M. Fontes and S. M. Homayouni, “Joint production and transportation scheduling in flexible manufacturing systems,” *J. Glob. Optim.*, pp. 1–30, 2018.
[59] P. Lacomme, M. Larabi, and N. Tchernev, “Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles,” *Int. J. Prod. Econ.*, vol. 143, no. 1, pp. 24–34, 2013.
[60] M. H. F. Bin Md Fauadi and T. Murata, “Makespan minimization of machines and Automated Guided Vehicles schedule using Binary Particle Swarm Optimization BT - International MultiConference of Engineers and Computer Scientists 2010, IMECS 2010, March 17, 2010 - March 19, 2010,” vol. III, pp. 1897–1902, 2010.
[61] L. Deroussi and S. Norre, “Simultaneous scheduling of machines and vehicles for the flexible job shop problem Solution approach : basic ideas,” *Int. Conf. Metaheuristics Nat. Inspired Comput.*, no. February, pp. 2–3, 2010.
[62] Q. Zhang, H. Manier, and M.-A. Manier, “Metaheuristics for Job Shop Scheduling with Transportation,” in *Metaheuristics for Production Scheduling*, 2013, pp. 465–493.
[63] M. V. S. Kumar, R. Janardhana, and C. S. P. Rao, “Simultaneous scheduling of machines and vehicles in an FMS environment with alternative routing,” *Int. J. Adv. Manuf. Technol.*, vol. 53, no. 1–4, pp. 339–351, 2011.
[64] C. Sahin, M. Demirtas, R. Erol, A. Baykasoğlu, and V. Kaplanoğlu, “A multi-agent based approach to dynamic scheduling with flexible processing capabilities,” *J. Intell. Manuf.*, vol. 28, no. 8, pp. 1827–1845, 2017.
[65] J. T. Lin, C. C. Chiu, and Y. H. Chang, “Simulation-based optimization approach for simultaneous

scheduling of vehicles and machines with processing time uncertainty in FMS," *Flex. Serv. Manuf. J.*, pp. 1–38, 2017.
[66] L. Deroussi, "A Hybrid PSO Applied to the Flexible Job Shop with Transport," in *Swarm Intelligence Based Optimization*, 2014, pp. 115–122.
[67] H. E. Nouri, O. Belkahla Driss, and K. Ghédira, "Solving the flexible job shop problem by hybrid metaheuristics-based multiagent model," *J. Ind. Eng. Int.*, vol. 14, no. 1, pp. 1–14, 2018.
[68] S. M. Homayouni and D. B. M. M. Fontes, "Joint scheduling of production and transport with alternative job routing in flexible manufacturing systems," *AIP Conf. Proc.*, vol. 2070, no. January, 2019.
[69] L. Mönch, "Autonome und kooperative steuerung komplexer produktionsprozesse mit multi-agenten-systemen," *Wirtschaftsinformatik*, vol. 48, no. 2, pp. 107–119, 2006.
[70] O. Sobeyko, *Integrated Process Planning and Scheduling in Flexible Job Shops*. deposit Hagen, 2018.
[71] U. Brecht, *BWL für Führungskräfte: Was Entscheider im Unternehmen wissen müssen*. Springer Fachmedien Wiesbaden, 2012.
[72] Y. Zheng, Y. Xiao, and Y. Seo, "A tabu search algorithm for simultaneous machine/AGV scheduling problem," *Int. J. Prod. Res.*, vol. 52, no. 19, pp. 5748–5763, 2014.
[73] I. A. Chaudhry, S. Mahmood, and M. Shami, "Simultaneous scheduling of machines and automated guided vehicles in flexible manufacturing systems using genetic algorithms," *J. Cent. South Univ. Technol.*, no. October 2011, 2011.
[74] S. Huang, "Optimization of Job Shop Scheduling with Material Handling by Automated Guided Vehicle," *ProQuest Diss. Theses*, p. 92, 2018.