Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 180 (2021) 825–832

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Industry 4.0 and Smart Manufacturing

Heuristic approaches for scheduling jobs and vehicles in a cyclic flexible manufacturing system

Martin Gutjahr[a,*], Hans Kellerer[b], Sophie N. Parragh[a]

[a]*Institute of Production and Logistics Management, Johannes Kepler University Linz, Altenbergerstraße 69, 4040 Linz, Austria*
[b]*Department of Statistics and Operations Research, University of Graz, Universitätsplatz 3, 8010 Graz, Austria*

Abstract

This paper addresses the scheduling of automated guided vehicles (AGVs) in a cyclic flexible flow shop environment. The vehicles travel along a single loop. All production machines are located alongside the track in the required order, with a possibility for multiple machines per stage. All AGVs are to be scheduled for a specific starting time and will then continuously circle the track. Pickup and delivery times are included in the travel time of a vehicle, stops are forbidden. Jobs may start upon arrival if their predecessor has been started for processing. Therefore, job completion times are dynamic. The considered objectives are the minimization of the number of AGVs and of the total makespan. For the regarded problem, different local search variants are proposed. Optimal results are produced using a brute force enumeration algorithm. Finally, fixed permutation schedules are compared to processing jobs according to a first-come-first-serve rule.

Keywords: Scheduling; Manufacturing; Transportation; Automated Guided Vehicles

1. Introduction

Manufacturing systems have been vastly growing in size over the past decades. Research focused on the subject faces the additional difficulty of a need for transportation between workstations. Sophisticated new systems are being developed to reduce the required amount of manual labour and to increase the production speed and volume. Automated guided vehicles (AGVs) have been a topic of interest for many years, because they may provide a satisfactory solution to this issue. These self-driving vehicles may transport jobs from one station to the next along a predefined route. Our research focuses on a specific subgroup of AGV scheduling problems known as cyclic scheduling. All jobs require subsequent processing in the same order on the machines. This problem is known as flow shop problem (FSP). For the regarded AGV scheduling problem, this allows us to assign the vehicles along a single route. Vehicles are pre-

* Corresponding author. Tel.: +43-732-2468-5504 ; fax: +43 732 2468 5514.
E-mail address: martin.gutjahr@jku.at

1877-0509
10.1016/j.procs.2021.01.332

defined to loop the track, starting and finishing a cycle at the depot. To circumvent bottlenecks at specific machines, we consider a flexible FSP. At each production stage, multiple machines are available to enable parallel processing. Our approach to the problem is motivated by related research concerning the minimization of the number of AGVs required to fulfil a given schedule of jobs on a given set of machines [1]. The problem has since been proven to be NP-hard even for the case with only a single vehicle [2]. We reformulate the initial problem to be multi-objective. Our goal is to minimize the overall makespan and the number of AGVs simultaneously. For our heuristic approach, we propose two local search based heuristics. Local search based algorithms have been successfully implemented for similar scheduling problems [3]. We then compute the Pareto frontier to display the trade off between the number of AGVs and the overall makespan. Solutions are compared to optimal solutions obtained using a brute force enumeration algorithm. Our newly developed, multidimensional heuristic approaches balance cost and benefit of transport vehicles and may prove especially valuable in problem settings where production speed increases revenue. The remainder of this paper is organised as follows. Section 2 reviews related work on AGV scheduling in manufacturing systems. Subsequently, the problem is formally defined in Section 3. Section 4 presents our solution approach. Benchmarks and results for test instances are discussed in Section 5. Conclusions and an indication of promising directions for future research are given at the end of the paper.

2. Related work

Various aspects of AGV scheduling problems have been studied in the last decades. The problem of simultaneous scheduling of machines and transport vehicles was first introduced in 1992 [4]. The proposed algorithm starts with the generation of a machine schedule, which is subsequently checked for feasibility with respect to the vehicle scheduling problem and was later improved by the same authors [5]. At the same time, Sabuncuoglu and Hommertzheim [6] published their research on the scheduling of AGVs and jobs on machines against the mean-flow time criterion. Various aspects of AGV systems have since been studied, such as design, control, and operational issues. For detailed literature on approaches in general the reader is referred to the works of Vis [7] and Ganesharajah, Hall, and Sriskandarajah [8].

The initial problems have been extended in various directions. Depending on the order in which jobs must be scheduled on different machines, the literature is divided into approaches for job shop problems (JSP) and flow shop scheduling. Related work for job shop environments introduces simultaneous scheduling and routing of transport vehicles to account for collision possibilities [9, 10, 11]. Furthermore, specific versions, such as optimizing the route between machines for a single vehicle [12] or allowing vehicle stops while disallowing an outgoing queue from machines [13] are studied. In case of FSPs, flexibility is often regarded as an additional factor of complexity. Surveys on flexible FSPs as well as hybrid FSPs can be found in the works of Wang [14] as well as Ruiz and Vázquez-Rodríguez [15]. The need for flexibility in the manufacturing process has been a topic of interest for many years [16]. Manufacturing systems that are able to respond to issues and change schedules when needed have been referred to as flexible manufacturing systems (FMS). The problem of simultaneous scheduling of transport vehicles and machines in FMSs has been addressed with the help of various heuristic approaches [17, 18].

A lot of research has focused on solving FSPs while regarding multi-objective optimization. As the FSP is known to be NP-hard in the single-objective case, most research has focused on the development of heuristics such as local search variants [3, 19, 20] and genetic algorithms [21]. For exact approaches to multi-objective FSPs Branch&Bound is the most common approach [22, 23, 24]. However, some problem tailored approaches, such as an exact parallel method based on the two-phase method [25], have been published. A survey on heuristic as well as exact approaches can be found in the paper by Minella, Ruiz and Ciavotta [26].

This paper focuses on a specific variant of AGV scheduling, which becomes particularly useful when considering FSPs. Blazewicz et al. [27] were among the first to introduce a cyclic scheduling problem. Parallel machines offer flexibility at each stage of the production process, while AGVs cycle a given route to deliver jobs between the stages. Since then, a vast number of researchers has examined cyclic scheduling in flow shop environments. Extensive surveys have been published on research done on the topic [28, 29, 30]. Differing from the approach generally used in the literature, we focus specifically on the scheduling of the automated transportations vehicles. Blazewicz et al. introduced the idea of minimizing the number of vehicles to fit a given schedule of jobs on machines [1]. We extend the regarded problem to remove any predefined scheduling constraints of jobs on machines in favour of general scheduling rules.

Furthermore, we reformulate the problem to be multi-objective, minimizing the number of AGVs as well as the overall makespan.

The most recent research focused on the problem of FMS by Beezão et al. [31] combines the scheduling of parallel machines with the limitation of tooling constraints. Demesure et al. [32] consider an AGV scheduling problem including decentralised motion planning. They introduce a two-step procedure to ensure collision free routing of AGVs. Furthermore, exact approaches for the simultaneous scheduling of machines and transport vehicles have been proposed [33].

3. Problem description

In the problem addressed in this paper, AGVs are to be scheduled to cycle a given loop while transporting jobs from machine to machine. All AGVs will move alongside the track consistently and without interruption. Loading and unloading operations are considered to be done in passing. Waiting for a job to be finished is prohibited. All jobs have to be processed at all machine stages in the predetermined order. Each stage of production may have multiple machines in parallel, which adds flexibility to the scheduling of the jobs. The problem therefore falls into the category of flexible flow shop. Jobs start their journey at the depot, where they are picked up by an AGV and transported to the first stage. After being processed at all stages, one last transport ensures that the finished job is brought back to the depot. For all machine stages, there is an incoming and an outgoing queue. Jobs delivered by AGVs wait in the incoming queue for an empty machine, while jobs in the outgoing queue await transport to the next station. The objective of the problem is to minimize the overall makespan, while minimizing the number of AGVs in the system. Consistent with the nomenclature used in related literature [12, 28], the considered problem is defined as a cyclic flexible robotic flow shop with many robots and buffering (CFRFS-MRB) and will further on be referred to as such.

As all AGVs are predefined to be going around the circuit at a constant speed, the location of an AGV at any point in time will therefore be determined by its starting time. We define time to be discrete with possible starting times equal to the time required by an AGV for a full lap. Let $Pos_{v,t}$ be the position of the vehicle v at a given point in time t and let T_v be the starting time of vehicle v with l denoting the lap time. Then,

$$Pos_{v,t} = (t - T_v) \mod l \qquad \forall t \geq T_v$$

holds. All AGVs are assumed to be able to start at the same time, simply providing capacity for multiple jobs. If an AGV was to start after more than l units of time, it may simply start l units of time earlier and complete a full lap to return to the depot at the initial starting time for the second lap. All jobs start at the depot and are considered to be in the system until an AGV has transported them from the last machine back to their origin. Each job j needs to be processed at each machine stage m for a specific amount of time p_{jm}. The completion time C_j for a given job j is defined as the number of units of time that passed from the start until the job has been returned to the depot after being processed at each stage.

We regard two objectives concurrently, the minimization of the makespan $C_{max} = \max_j\{C_j\}$ and the minimization of the number of used AGVs k. As we set an upper limit for the number of vehicles K, we minimize C_{max} for all different $k \in \{1, \ldots, K\}$. During the optimization, the objective value C_{max} is obtained via a simulation of the problem for given starting times $T = \{T_1, \ldots, T_v, \ldots, T_k\}$ with T_v being the starting time of vehicle v. The main constraints to be considered by the simulation are capacity constraints of the AGVs as well as the capacity of each machine at a stage, both of which are considered to be one. The simulation returns C_{max}, all assignments to a specific machine A_{jm} for job j at machine stage m, and all assignments of an AGV D_{jm} to transport job j from stage m to its successor stage.

We introduce two different scheduling rules for the scheduling of jobs on machines. The first approach follows a permutation schedule, with all jobs requiring scheduling at the machines in the same order as they left the depot. For the automated pickup of jobs by AGVs, this entails that each vehicle always loads the job in the outgoing queue that is scheduled next in line for processing at the next stage. The second approach introduces a first-come-first-serve (FCFS) rule. All jobs start in a predefined order from the depot; however, jobs may overtake others at the machine stages simply by being completed first at a parallel machine. The assignment of jobs to machines will therefore follow the order of the incoming queue. The same rule is set for the outgoing queue and pick up by AGVs respectively.

4. Solution approach

Multi-objective scheduling problems are well known to be computationally hard to solve. While some progress was made on exact approaches to solve related problems, most work has proven to be non-competitive in terms of runtime for larger instances. Some authors even provided competitive heuristic approaches in the introductory paper of their exact approach[22]. We therefore focused our choice of solution approach on metaheuristics that could take advantage of the structure of a solution. Motivated by their success in the scheduling domain and more precisely for multi-objective FSPs [3, 19] as well as flexible flow shop approaches [34], we develop two neighbourhood search based algorithms. Our first implementation is an iterated local search (ILS). ILS takes a problem tailored local search algorithm and applies it to the solution repeatedly. Starting from an initial solution, the neighbourhood is checked for the best immediate improvement. The neighbourhood of a solution is considered to be all assignments that can be reached by changing the starting time of a single vehicle by one unit. Each descent to a local optimum is counted as one iteration. If no improvement can be found, the ILS approach introduces a perturbation step. In this step, the algorithm tries to escape the current local optimum by changing the solution to be a significant distance away from the last. Then, the next iteration starts by applying the local search algorithm. The functionality of our implementation is shown in algorithm 1.

Algorithm 1 Iterated local search

```
for AGVS in range(maxAGVS): do
    currentsol = runconstruction(AGVS)
    for i in range(iterations): do
        localsearch(currentsol, AGVS, 1)
        while improvement_found do
            localsearch(currentsol, AGVS, 1)
        end while
        if currentsol < bestsol then:
            store_sol(AGVS)
        end if
        currentsol = perturbation(currentsol)
    end for
end for
```

The second approach is a variable neighbourhood search (VNS). This local search based approach tries to take advantage of changing the neighbourhood in between iterations [35]. Similar to the ILS, a local search algorithm is applied to find the best improvement in the neighbourhood. However, if no improvement can be found, then the VNS changes the neighbourhood it searches. The analysed neighbourhood will move further and further from the location of the initial one, until an improvement can be found once more. Then, the algorithm returns to the immediate neighbourhood of the new solution. The pseudocode of our implementation is depicted in Algorithm 2. Instead of changing the starting time of only single vehicle at a time, the VNS changes the number of AGVs currently allowed to move *cAGV* in between iterations. If no improvement could be found, the neighbourhood is increased in size by increasing *cAGV* by one, therefore allowing an additional change in starting time in a single step. *cAGV* is reset to one as soon as another improvement was found.

Our implementation of the local search used in the ILS and VNS approaches takes advantage of time being regarded in discrete intervals. Every application of the local search checks the neighbourhood of the incumbent solution that can be reached by changing the starting time of any AGV by a single unit of time. The size of the neighbourhood is therefore twice the number of AGVs in the current solution. At least one AGV has to start at time 0. Any waiting time at the beginning of the model will simply incur an overall delay of that length in comparison to a schedule of all AGVs starting that amount of time earlier.

In order to produce optimal results we take advantage of the discretization of time once again. A finite amount of possible starting times allows us to do a complete enumeration of possible combinations of the starting times for a

Algorithm 2 Variable neighbourhood search

```
for AGVS in range(maxAGVS): do
    currentsol = runconstruction(AGVS)
    cAGV = 1
    for i in range(iterations): do
        localsearch(currentsol, AGVS, cAGV)
        while improvement_found do
            localsearch(currentsol, AGVS, cAGV)
            cAGV = 1
        end while
        if currentsol < bestsol then:
            store_sol(AGVS)
        end if
        cAGV += 1
    end for
end for
```

given number of AGVs. We therefore implement a brute force algorithm to produce optimal solutions for benchmarking purposes.

5. Benchmarks and results

The local search algorithms described in the previous chapters have been implemented in Python and run against a set of benchmark instances generated artificially. Instances vary in the number of jobs as well as the number of machines. The number of machines is varied based on the performance of our heuristics. Smaller instances start at 10 machine stages and take a couple of minutes to be solved, while the larger instances go up to 40 machine stages and 12 hours of runtime. Smaller instances contain 25 or 50 jobs, while larger instances include 100 or 250. All tests are run on two Intel Xeon Quad-Core CPU @2.93GHz with 48GB of RAM. Due to the runtime of the brute force algorithm increasing exponentially with the instance size, a wall time of 48 hours has been set.

Overall, we compare four different approaches. For both ILS and LNS, we introduce two approaches for the handling of jobs for processing at the machines, a permutation schedule and a FCFS variant. As a higher number of available AGVs yields an ever decreasing amount of time saved, the maximum number of AGVs has been limited to 8. The heuristic approaches are used to generate solutions for all possible numbers of AGVs up to the set limit. The solutions are then compared to the optimal solutions for those instances that could be solved within the time limit.

To analyse and compare the performance of the considered approaches concerning the two objectives, minimization of the number of vehicles and minimization of the overall makespan, we calculate the hypervolume indicator. This indicator calculates the hypervolume of the dominated portion of the objective space as a measure for the quality of the generated Pareto set [36]. With our two objectives being the minimization of the number of vehicles as well as the minimization of the maximum makespan for the jobs, the reference point can be directly calculated. On the X-Axis, referencing the number of vehicles, we set the value to the highest number we allow in our model. On the Y-Axis, the referencing point takes the value of the makespan achieved using just a single vehicle.

The hypervolume-values obtained for each of the instances that were solved to optimality are given in Table 1 together with the respective runtimes. They are given in percentages of the hypervolume-value (HV%) of the optimal solution. Table 2 depicts the absolute hypervolume-values (HV) of the four heuristic approaches for all those instances for which the brute-force enumeration algorithm was not able to solve the problem within the time limit. In both tables the characteristics of the regarded instance are given in the leftmost column, with the first value denoting the number of machine stages and the second representing the number of jobs in the system. The results show that VNS performs comparably to ILS; however, VNS is considerably faster. In addition, implementing a FCFS strategy performs similar to forcing a permutation schedule, but reduces computation time as well.

Table 1. Instances solved to optimality. Values shown depict the hypervolume-values in percentage (HV%) of the optimal solution for each of the four combinations between the considered scheduling rules and the proposed heuristic approaches. Furthermore, the runtime of each approach is given.

	Permutation				FCFS			
	ILS		VNS		ILS		VNS	
Instance	HV%	runtime	HV%	runtime	HV%	runtime	HV%	runtime
10/25	99.63%	0:01:33	99.63%	0:00:56	99.42%	0:01:10	99.31%	0:00:44
10/50	99.66%	0:03:26	99.68%	0:01:59	99.78%	0:02:03	99.76%	0:01:09
10/100	99.95%	0:08:19	99.92%	0:04:42	99.85%	0:03:13	99.92%	0:02:08

Table 2. Instances without optimal solution. Values shown depict the hypervolume-values (HV) for each of the four combinations between the considered scheduling rules and the proposed heuristic approaches. Furthermore, the runtime of each approach is given.

	Permutation				FCFS			
	ILS		VNS		ILS		VNS	
Instance	HV	runtime	HV	runtime	HV	runtime	HV	runtime
10/250	23972	0:37:02	23951	0:19:51	23995	0:07:43	24001	0:04:08
20/25	7362	0:06:51	7344	0:04:38	7354	0:04:21	7334	0:03:03
20/50	11907	0:12:52	11883	0:08:21	11845	0:07:48	11836	0:05:06
20/100	20982	0:27:45	21004	0:19:20	21013	0:15:51	21031	0:09:24
20/250	48680	2:57:33	48699	1:56:47	48714	0:33:58	48706	0:22:05
30/25	14098	0:23:12	14042	0:13:23	14119	0:12:40	13995	0:08:12
30/50	20441	0:41:16	20800	0:22:40	20817	0:25:42	20510	0:16:03
30/100	34280	1:40:58	33990	0:44:21	34004	0:39:10	34038	0:24:22
30/250	74906	6:03:22	74882	4:24:43	74916	1:54:01	75340	1:26:44
40/25	22415	0:58:29	22505	0:36:28	22708	0:25:44	22485	0:23:18
40/50	31205	2:05:14	31156	1:00:54	31247	1:08:01	30659	0:39:23
40/100	48978	3:32:38	48896	2:28:05	49145	2:00:21	49096	0:58:01
40/250	103493	11:55:43	102975	7:49:50	103484	5:00:22	102880	3:20:11

Further testing was done to check the performance of the proposed methods for higher numbers of iterations. The initial setting of 20 iterations was compared to 60, 40, 80, and 100 iterations. Figure 1 shows Pareto frontiers for varying numbers of iterations including AGV numbers of five and above for the instance considering 10 machine stages and 100 jobs. Table 3 contains the runtimes for runs with different numbers of iterations. The computation time grows almost linearly with the number of iterations, with small deviations occurring due to slightly varying iteration lengths. Large instances show that the bottleneck responsible for longer runtimes comes from the requirement to run a full simulation of the problem to obtain the solutions for a specific set of starting times.

Table 3. Hyper-volume values and runtimes for increasing numbers of iterations for each of the four combinations between the considered scheduling rules and the proposed heuristic approaches. All values are generated for the instance including 10 machine stages and 100 jobs.

	Permutation				FCFS			
	ILS		VNS		ILS		VNS	
Iterations	HV	runtime	HV	runtime	HV	runtime	HV	runtime
20	9736	0:08:19	9733	0:04:42	9760	0:03:13	9767	0:02:08
40	9739	0:15:50	9738	0:09:54	9766	0:06:29	9767	0:03:56
60	9739	0:24:29	9739	0:14:23	9770	0:10:20	9768	0:05:44
80	9739	0:33:04	9739	0:19:46	9770	0:13:21	9768	0:07:25
100	9739	0:40:20	9739	0:23:14	9771	0:17:23	9770	0:09:37

6. Conclusion

Implementing the proposed heuristic approaches and comparing them with a brute-force enumeration algorithm shows that solutions of high quality are achieved rather quickly for smaller instances. As the problem size grows, an increase in the deviation from the optimal solution as well as significantly longer runtimes occur. Our results show that the proposed VNS approach is generally preferable to ILS that does not increase the size of the neighbourhood.

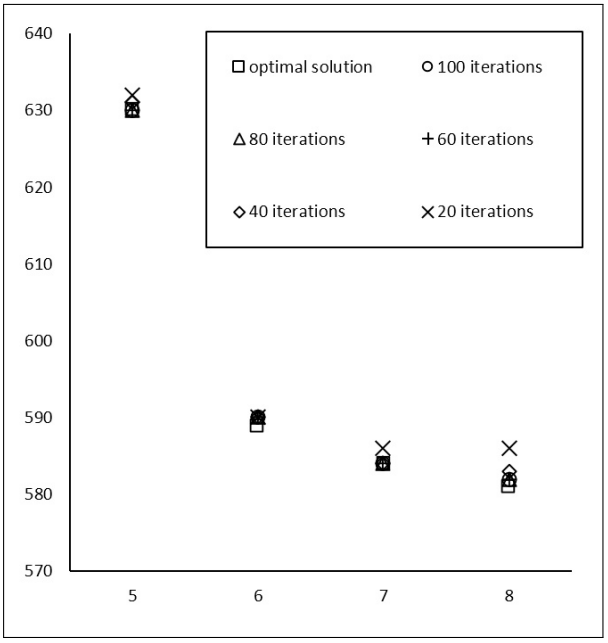

Fig. 1. Objective values for the minimization of the makespan achieved with increasing numbers of iterations for the instance including 10 machine stages and 100 jobs.

Future research will involve the development of a tailored exact approach so as to assess the quality of the proposed heuristics also on larger instances. Furthermore, the problem offers additional flexibility to increase complexity, for instance, allowing general changes of the order of the jobs even at the depot stage. We will further look into the generation of the best schedule for jobs on machines to fit the AGV starting times, which increases the problem complexity considerably.

7. Acknowledgments

This work has been partially funded by the Austrian Science Fund (FWF): P 31366

References

[1] Blazewicz, Jacek et al. (1997) "Scheduling vehicles in a cyclic flexible flowshop." *Journal européen des systèmes automatisés* **32** (4): 441–451.
[2] Espinouse, Marie-Laure, Grzegorz Pawlak, and Malgorzata Sterna. (2017) "Complexity of scheduling problem in single-machine flexible manufacturing system with cyclic transportation and unlimited buffers." *Journal of Optimization Theory and Applications* **173** (3): 1042–1054.
[3] Arroyo, José Elias Claudio, and Vinícius Amaral Armentano. (2005) "Genetic local search for multi-objective flowshop scheduling problems." *European Journal of Operational Research* **167** (3): 717–738.
[4] Ulusoy, Gündüz, and Ümit Bilge. (1992) "Simultaneous scheduling of machines and material handling system in an FMS." *IFAC Proceedings Volumes* **25** (8): 15–25.
[5] Bilge, Ümit, and Gündüz Ulusoy. (1995) "A time window approach to simultaneous scheduling of machines and material handling system in an FMS." *Operations Research* **43** (6): 1058–1070.
[6] Sabuncuoglu, Ihsan, and Don L. Hommertzheim. (1992) "Experimental investigation of FMS machine and AGV scheduling rules against the mean flow-time criterion." *The International Journal of Production Research* **30** (7): 1617–1635.
[7] Vis, Iris FA. (2006) "Survey of research in the design and control of automated guided vehicle systems." *European Journal of Operational Research* **170** (3): 677–709.
[8] Ganesharajah, Tharma, Nicholas G. Hall, and Chelliah Sriskandarajah. (1998) "Design and operational issues in AGV-served manufacturing systems." *Annals of Operations Research* **76**: 109–154.
[9] Emde, Simon, and Nils Boysen. (2012) "Optimally routing and scheduling tow trains for JIT-supply of mixed-model assembly lines." *European Journal of Operational Research* **217** (2): 287–299.

[10] Lacomme, Philippe, Mohand Larabi, and Nikolay Tchernev. (2013) "Job-shop based framework for simultaneous scheduling of machines and automated guided vehicles." *International Journal of Production Economics* **143** (1): 24–34.
[11] Nouri, Houssem Eddine, Olfa Belkahla Driss, and Khaled Ghédira. (2016) "Simultaneous scheduling of machines and transport robots in flexible job shop environment using hybrid metaheuristics based on clustered holonic multiagent model." *Computers & Industrial Engineering* **102**: 488–501.
[12] Kats, Vladimir and Eugene Levner. (1997) "A strongly polynomial algorithm for no-wait cyclic robotic flowshop scheduling." *Operations Research Letters* **21** (4): 171–179.
[13] Kats, Vladimir and Eugene Levner. (1997) "Minimizing the number of robots to meet a given cyclic schedule." *Annals of Operations Research* **69**: 209–226.
[14] Wang, Hong. (2005) "Flexible flow shop scheduling: optimum, heuristics and artificial intelligence solutions." *Expert Systems* **22** (2): 78–85.
[15] Ruiz, Rubén, and José Antonio Vázquez-Rodríguez. (2010) "The hybrid flow shop scheduling problem." *European Journal of Operational Research* **205** (1): 1–18.
[16] Sethi, Andrea Krasa, and Suresh Pal Sethi. (1990) "Flexibility in manufacturing: a survey." *International Journal of Flexible Manufacturing Systems* .**2** (4): 289–328.
[17] Reddy, B. S. P., and C. S. P. Rao. (2006) "A hybrid multi-objective GA for simultaneous scheduling of machines and AGVs in FMS." *The International Journal of Advanced Manufacturing Technology* **31** (5-6): 602–613.
[18] Zheng, Yan, Yujie Xiao, and Yoonho Seo. (2014) "A tabu search algorithm for simultaneous machine/AGV scheduling problem." *International Journal of Production Research* **52** (19): 5748–5763.
[19] Gupta, Jatinder ND, Karsten Hennig, and Frank Werner. (2002) "Local search heuristics for two-stage flow shop problems with secondary criterion." *Computers & Operations Research* **29** (2): 123–149.
[20] Schulz, Sven, Janis S. Neufeld, and Udo Buscher. (2019) "A multi-objective iterated local search algorithm for comprehensive energy-aware hybrid flow shop scheduling." *Journal of Cleaner Production* **224**: 421–434.
[21] Ishibuchi, Hisao, and Tadahiko Murata. (1998) "A multi-objective genetic local search algorithm and its application to flowshop scheduling." *IEEE Transactions on Systems, Man, and Cybernetics, part C (Applications and Reviews)* **28** (3): 392–403.
[22] Toktaş, Berkin, Meral Azizoğlu, and Suna Kondakcı Köksalan. (2004) "Two-machine flow shop scheduling with two criteria: Maximum earliness and makespan." *European Journal of Operational Research* **157** (2): 286–295.
[23] Basseur, Matthieu, et al. "Cooperation between branch and bound and evolutionary approaches to solve a bi-objective flow shop problem." (2004) *International Workshop on Experimental and Efficient Algorithms*: 72–86.
[24] Lin, B. M. T., and J. M. Wu. (2006) "Bicriteria scheduling in a two-machine permutation flowshop." *International Journal of Production Research* **44** (12): 2299–2312.
[25] Lemesre, Julien, Clarisse Dhaenens, and El-Ghazali Talbi. (2007) "An exact parallel method for a bi-objective permutation flowshop problem." *European Journal of Operational Research* **177** (3) : 1641–1655.
[26] Minella, Gerardo, Rubén Ruiz, and Michele Ciavotta. (2008) "A review and evaluation of multiobjective algorithms for the flowshop scheduling problem." *INFORMS Journal on Computing* **20** (3): 451–471.
[27] Blazewicz, Jacek, et al. (1991) "Scheduling tasks and vehicles in a flexible manufacturing system." *International Journal of Flexible Manufacturing Systems* **4** (1): 5–16.
[28] Crama, Yves, et al. (2000) "Cyclic scheduling in robotic flowshops." *Annals of Operations Research* **96** (1-4): 97–124.
[29] Qiu, Ling, et al. (2002) "Scheduling and routing algorithms for AGVs: a survey." *International Journal of Production Research* **40** (3): 745–760.
[30] Levner et al. (2010) "Complexity of cyclic scheduling problems: A state-of-the-art survey." *Computers & Industrial Engineering* **59** (2): 441–451.
[31] Beezão, Andreza Cristina, et al. (2017) "Scheduling identical parallel machines with tooling constraints." *European Journal of Operational Research* **257** (3): 834–844.
[32] Demesure, Guillaume, et al. (2017) "Decentralized motion planning and scheduling of AGVs in an FMS." *IEEE Transactions on Industrial Informatics* **14** (4): 1744–1752.
[33] Fontes, Dalila BM M., and Seyed Mahdi Homayouni. (2019) "Joint production and transportation scheduling in flexible manufacturing systems." *Journal of Global Optimization* **74** (4): 879–908.
[34] Negenman, Ebbe G. (2001) "Local search algorithms for the multiprocessor flow shop scheduling problem." *European Journal of Operational Research* **128** (1): 147–158.
[35] Mladenović, Nenad, and Pierre Hansen. (1997) "Variable neighborhood search." *Computers & Operations Research* **24** (11): 1097–1100.
[36] Zitzler, Eckart, Dimo Brockhoff, and Lothar Thiele. (2007) "The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration." *International Conference on Evolutionary Multi-Criterion Optimization*: 862–876.