Advances in Science and Technology Research Journal

Advances in Science and Technology Research Journal 2023, 17(3), 280–293
https://doi.org/10.12913/22998624/166588
ISSN 2299-8624, License CC-BY 4.0

Received: 2023.04.19
Accepted: 2023.05.19
Published: 2023.06.01

Constraint Programming for Flexible Flow Shop Scheduling Problem with Repeated Jobs and Repeated Operations

Kateryna Czerniachowska[1], Radosław Wichniarek[2*], Krzysztof Żywicki[2]

[1] Wroclaw University of Economics and Business, Wroclaw, Poland
[2] Poznan University of Technology, Poznan, Poland
* Corresponding author's e-mail: radoslaw.wichniarek@put.poznan.pl

ABSTRACT
The manufacturing process relies heavily on production scheduling to guarantee effective production schedules, reduce costs and product shortages, and get away from delays, interruptions, and waste products. Scheduling issues have been effectively solved via constraint programming. In this paper, we consider a constraint programming formulation of the flexible flow shop scheduling problem with repeated jobs and operations repetitions. We also implemented seven fast heuristics and compared the results with the constraint programming solution. The results from using the CPLEX solver as a solution tool were satisfactory. Computational experiments demonstrate that using constraint programming can be significantly more advantageous than using heuristics.

Keywords: flexible flow shop scheduling problem, constraint programming, heuristics

INTRODUCTION

Due to Industry 4.0, also known as the fourth industrial revolution, which aspires to fully automate and digitize processes, particularly production processes, scheduling has become more crucial than ever in the industrial setting [1,2]. Technology advancements, including the Internet of Things, big data, electric vehicles, 3D printing, cloud computing, artificial intelligence, and cyber-physical systems, are included in Industry 4.0 [3].

For manufacturing companies aiming to advance their production facility, production scheduling has become a requirement. The distribution of the operations, processes, and resources needed to produce goods and services is referred to as production scheduling.

Due to the introduction of "Internet of Things" applications that present numerous opportunities for improving existing systems, this is currently becoming an even more significant application field for scheduling theory, to the point that the modern world is frequently referred to as the start of the fourth industrial revolution [1,4].

For expanding manufacturing enterprises to increase their output, production planning is crucial. In manufacturing, scheduling is used to inform a production facility when to manufacture something, with whom, and on what machinery in order to meet customer deadlines while also reducing production time and costs. In order to maximize productivity, make the most of the available resources, and cut expenses, production scheduling strives to do all three.

A situation where machines can communicate with one another in the most effective way possible, "free" from the error-prone interaction of human staff, is becoming more and more likely as facilities become more automated. This is already true in logistics, as seen by Amazon's autonomous shelf robots in their warehouses and automated packaging machines that are five times faster than people [5], as well as in transportation, where it is predicted that highways would be safer when free of human-driven cars [6]. Scheduling theory developments can now more than

ever have a significant impact on the real-world economy [7,8].

In traditional scheduling issues, decisions about technology activities and transport duties are frequently made separately, or issues are only present with technological chores. The production schedule's quality suffers as a result of this simplification. Moreover, sometimes, situations in industrial production processes occur where setup time is much longer than processing time. Additionally, there are not many transport resources, which could lead to a stalemate in the system [9].

Some businesses use Gantt charts, data from their ERP system, and Excel spreadsheets to manage schedules manually. Companies may reach a point when it becomes extremely challenging to keep manually scheduling their factory using Excel spreadsheets and data entry due to constraints on staff and consumer demand.

The goal of the production scheduling is to arrange resources effectively and affordably to fulfil customer orders. Planning production improves the productivity of the factory.

Due to their numerous industrial applications, shop scheduling issues are one of the most studied optimization issues. Variants of the flow shop scheduling problem and the job shop scheduling problem have drawn much attention from production scheduling academics in recent years. The well-known flexible flow shop scheduling problem (FFSSP) serves as the foundation for the issue being discussed in this article.

Constraint programming (CP) is an effective method for addressing combinatorial production planning issues. The search space is narrowed using sophisticated deduction, and the search space is explored using a combination of variable- and value-selection heuristics. Utilizing a constraint programming method, the production FFSSP problem could be resolved.

BACKGROUND

The job shop scheduling problem (JSSP) is frequently regarded as one of the most challenging problems in combinatorial optimization [9-12]. The issue with the goal criteria, which is the completion time of all activities, has been in-depth analyzed by numerous researchers due to its enormous practical significance, and numerous precise as well as approximative methods have been devised [9].

A selection of jobs and a set of machines are provided in the job shop problem. There is no recirculation and each job must be completed by a number of machines in a specific order. Each job consists of a set of operations (tasks) that must be completed by a machine designated for each operation (task) in a predetermined length of time, uninterrupted. Each task has a stated execution time, which often includes both the time required to set up the workstation for the task at hand and the time required to return it to its pre-task condition. It frequently also accounts for travel time to the created item place. The device is capable of carrying out up to one operation at any given time. The goal is to reduce the makespan (i.e., complete production ends as fast as possible). This issue involves creating a schedule using a predetermined priority rule (assigning related jobs to the appropriate machines within a specified time frame). The above-described issue is a classic combinatorial optimization issue and is a member of the problem class that is strongly NP-hard [9].

These days, just-in-time delivery poses particular difficulties for the supplier sector. With just-in-time delivery, the supplier is required to give the client the material when it is required. The present demand of the customer's production and assembly unit determines delivery schedules, e.g., the automotive industry [13].

A job shop problem is used by a manufacturer of automobile components to produce a wide range of parts. The setups for many manufacturing procedures are extensive. The manufacturer must rapidly determine whether it has the capacity necessary to take an order when a customer calls in.

In the paper [14] researchers studied the scheduling issue in job shop systems with transportation. Similar to a standard scheduling problem, it entails allocating a set of tasks (production and transportation tasks) to a set of resources (processing equipment, transportation vehicles) while minimizing the maximum completion time of a production order (makespan) and accounting for the related constraints — production and transportation constraints. Researchers adopted the Variable Neighborhood Search metaheuristic to reorganize various job scheduling in order to tackle transport and processing tasks. This metaheuristic used asynchronous local search techniques to discover the best resource task allocation while minimizing the makespan [14].

The study [9] provided a generalized job-shop problem that sets up machines under deadlock-free operation conditions while accounting for travel time between workstations. A multitude of autonomous guided vehicles are used in the automated transportation system. The optimization criterion was the average completion time of all jobs. This problem was solved using the developed computational application, in which the priority algorithms were used [9].

In the paper [15] researchers discussed a brand-new multimachine multiobjective task scheduling problem (P) that was modelled after the one Renault, the automaker, put up for the ROADEF 2005 Challenge. The latter issue's resource use was balanced by placing a major emphasis on minimizing the smoothing penalties. In (P), smoothing costs are also taken into consideration, but actual elements such as eligibility limits, non-identical parallel production lines/machines (instead of a single line), machine and job-dependant setup times and costs, and overall makespan reduction are also introduced. Assigning a reasonable order of importance to the goals is a method to minimize the overall makespan, smoothing costs, and setup costs [15].

The majority of advanced economies heavily rely on technology, with technology-intensive industries predominating. The majority of automated and routine jobs are now carried out by robots because cost and productivity are two of the key factors influencing manufacturing competitiveness. Flexibility is another important factor, and it may be improved by using resources that can carry out a variety of jobs, allowing for the use of various resources for various tasks. Higher levels of automation and greater flexibility produce extremely complex industrial processes that call for the best possible decision-making, especially in scheduling decisions, which entail the long-term distribution of finite resources [16].

The author of the paper [2] introduced various types of flexibility and investigated how the flexibility type, the quantity of permitted flexibility, and the presence of machine-dependent processing times impacted the solution quality that can be attained by a state-of-the-art constraint solver within a constrained time since the classic problem formulation is more general than what is found in most modern industrial settings. The difficulty can be lessened, according to the results, by particular types of flexibility, higher flexibility factors, and the absence of machine-dependent processing delays [2].

Another group of researchers [17] demonstrated the value of using a decision-making tool in an Industry 4.0 context to enhance resource management and reap the rewards of real-time data. They also proposed a task allocation optimization method based on a flexible job-shop problem that can be solved in a few minutes to dynamically allocate real-time tasks to operators in a 4.0 context. Next, they outlined the advantages of using a dynamic allocation system for tasks and resources requiring multiple skills that are released periodically and arbitrarily [17].

The authors of the paper [16] put forth the MILP model for the flexible job shop problem with transportation that solved benchmark examples with optimally small and some medium-sized sizes. They provided a heuristic that outperforms contemporary heuristics and can quickly locate decent solutions [16].

In multiobjective scheduling issues, decreasing the makespan while taking setup expenses and timeframes into account is common [15]. The flexible multiobjective job shop problem is proposed and solved by the authors of the paper [18] by combining particle swarm and local search techniques. In contrast to a job shop problem, where a set of jobs must be scheduled on a set of various machines, and each job must follow a specific routing on the machines, a flexible job shop problem is an extension where each job can be processed by any machine of a set along various routes, taking eligibility restrictions into consideration [15].

Due in part to the wide range of production environments it can model, the flow shop scheduling problem (FSP) is a sequencing problem that has drawn a lot of interest from experts and researchers in recent years [7].

A number of machines placed in a series must process a set of jobs or products in an FSP. All jobs must be processed via each machine in the same order. There is a defined processing time which is needed for each job in the machine. The main objective of FSP is to choose a processing order for the jobs that will optimize one or more efficiency metrics [19].

The demanding order in which all operations must be completed on every task is known as flow shop scheduling, which is a particular kind of job shop scheduling. Production facilities and

computer designs may both benefit from flow shop scheduling.

In the study [20], a two-machine flow shop problem with ineligible transportation lags in the production process was examined. Two different types of transportation were taken into consideration: one for moving partially completed jobs between machines and the other for delivering fully completed jobs to clients. Two heuristic algorithms are also created to effectively solve the issue - Johnson's rules order the tasks, and the First Only Empty algorithm decides how each transportation batch should be combined [20].

A set of parallel machines set up in a sequence of phases is related to an FFS scheduling problem. There are several identical machines running in parallel at each level. Each work can disregard one or more phases and does not need to be processed at every stage. A single job can only be processed by each machine at once.

A broad type of flow shop when two or more parallel machines are used for at least one step is known as a flexible flow shop (FFS), sometimes known as a multi-processor flow shop [12,21]. There are some tasks that must be completed in a sequence of steps in order to optimize the specified objective function. Although there have been numerous variations of this issue, they all share the following characteristics [22, 23]:

- There are certain jobs identified by that need processing time, such as a stage.
- There are at least two stages to the procedure.
- In each stage, there are parallel machines.
- From stage 1 to stage 2, the jobs should be processed using the same production flow. Nevertheless, some jobs may skip several steps.

The papers [22, 24] provided a thorough overview of the research on the hybrid flow shop scheduling problem. The latter summarized the literature on exact, heuristic, and metaheuristic problem-solving strategies. For each of the different versions of the problem, they took into account various assumptions, restrictions, and goal functions. The related works can be divided into two categories: the solution method and the hybrid flow-shop features and production constraints [25].

The hybrid flow shop scheduling problem with job family is examined in the research [26]. One or more parallel machines are present at each of the two or more production stages that make up a hybrid flow shop, which is an expanded version of the standard flow shop. The addition of parallel machines boosts flexibility and production. The authors proposed an approach that doesn't keep any machine idle during scheduling in a hybrid flow shop. Their study demonstrated how adding deliberate idle time to a non-delay schedule can cut setup time and makespan even more [26].

The hybrid flow shop scheduling problem with a rework is the subject of the paper [25]. With this issue, jobs are reviewed at the very end, and those that were improperly processed are sent back. The authors modelled a hybrid flow-shop with the re-entrant flow as a result of a work potentially visiting a stage more than once. Each work begins at stage 1 and ends at stage 0 in a normal hybrid flow shop scheduling problem, visiting each stage only once and being processed by one of the parallel machines at each level. In additional iterations of the problem, known as re-entrant hybrid flow shops, each job may go through the same step more than once. Many different businesses, such as the final inspection system in the manufacture of automobiles, may employ this type of shop. In order to address the issue of sequence-dependent setup times and unrelated parallel machines, the authors expanded a few heuristic methods based on a few fundamental dispatching rules and suggested a variable neighbourhood search [25].

Even in more conventional situations like metal machining, Industry 4.0 concepts allow for a new way of thinking about the allocation of human resources. While some manual operations still need to be carried out by operators, the milling of parts on Computer Numerical Control equipment is automated. The current strategy usually involves statically allocating operators to one or more computers. Thus, avoidable bottlenecks are created [17].

Finding the appropriate instruments requires management and the appropriate software for production planning and scheduling. To manage the flow and identify production schedule problems, efficient production planning software is required.

The complexity of the task (relationships between operations and additional constraints) and the size of the problem (the number of different orders and the number of available machines) are the main factors that determine how long an algorithm will take to process.

This issue might be resolved using the Integer Linear Programming (ILP) approach. By minimizing the objective function, ILP finds the best

solution to issues that can be solved by formulating a set of inequalities as constraints.

Especially for complicated problems that are difficult to solve with integer linear equations, constraint programming is a contemporary paradigm for combinatorial optimization issues [11]. Constraint programming first appeared in the field of artificial intelligence, but it has created high-quality outcomes when used to address issues with production sequencing [12].

A problem instance that has been defined in this way can be given to an expert solver. The number of tasks heavily influences the number of variables, and the number of equations influences the number of restrictions brought on by the difficulty of the assignment.

Nowadays, a lot of scheduling problems have been successfully addressed by constraint-based techniques [4, 10]. A constraint solver tool included in the IBM ILOG CPLEX Optimization Studio is called Constraint Programming Optimizer (CPO). ILOG was a well-known software business specializing in supply chain and optimization issues. The creation of a generic optimization suite, combining CPLEX to handle mixed integer linear programming (MILP) problems and CP Optimizer for CP problems, has received a lot of attention since ILOG, the maker of the well-known CPLEX mathematical programming tools, was acquired by IBM in 2008 [4, 28].

There are various possibilities available today for the manufacturing company to locate production planning software that has been created especially for contemporary manufacturers.

The authors of the paper [28] proposed a formulation for the order insertion problem using mixed integer linear programming, which avoided complicated Gantt chart manipulations while still ensuring feasibility. They put forth and thoroughly tested heuristics based on evolutionary algorithms, tabu search, simulated annealing, myopic search, and a relaxation of the shortest path [28].

Companies often use Manufacturing Execution System (MES) software. In this case, not only should scheduling be improved but also the entire production process as a whole. Data collected by effective MES software can show where inefficiencies are occurring. MES software provides visibility of inefficiencies such as machine utilization, waste of raw materials, downtime of machines, problems with the changes, and overall equipment effectiveness metrics.

Heuristics have largely been used by researchers to address the variants of scheduling problems. These heuristics may produce quick and efficient solutions, although they are frequently customized. Furthermore, because they merge the problem description and the solution method into one framework, the effectiveness of these techniques heavily depends on the correct implementation and careful tweaking of parameters. In contrast, the issue description and the solution approach are separated in the mathematical modelling approach [29]. Furthermore, practitioners have been able to create more intricate and sophisticated problems as computer hardware, and solvers have advanced [28].

PROBLEM DEFINITION AND FORMULATION

Our model is motivated by the electronic components production example. Many people view the production of electronics as a high-level manufacturing issue. An electronic manufacturer utilizing an original equipment manufacturer is the ideal illustration of a business that develops electronic items to sell or promote them. Effective interaction between the manufacturer and the consumer is necessary for electronic design. The manufacturer will frequently request some idea from the client or clients and then work on it.

Figure 1 presents an investigated problem. A company manufactures a wide variety of components. There are 6 production stations (machines). There are 2 positions (1 and 6) which are not replaceable. There are 4 universal production stations, mutually replaceable at each station, all of them can be implemented assembly/production tasks assembly of each component that is produced.

There are tasks which could be repeated. Each task consists of a set of operations. The operations could be repeated, but they must be executed on the same machine. Some operations could be executed only on the first or the last machine. The rest operations could be executed on the medium replaceable machine. An operation is characterized by the execution time on the machine. The transportation time between machines is neglected.

The supplier needs to decide how to assign the components' production operations (operations) to (machines) so that the makespan is minimized.

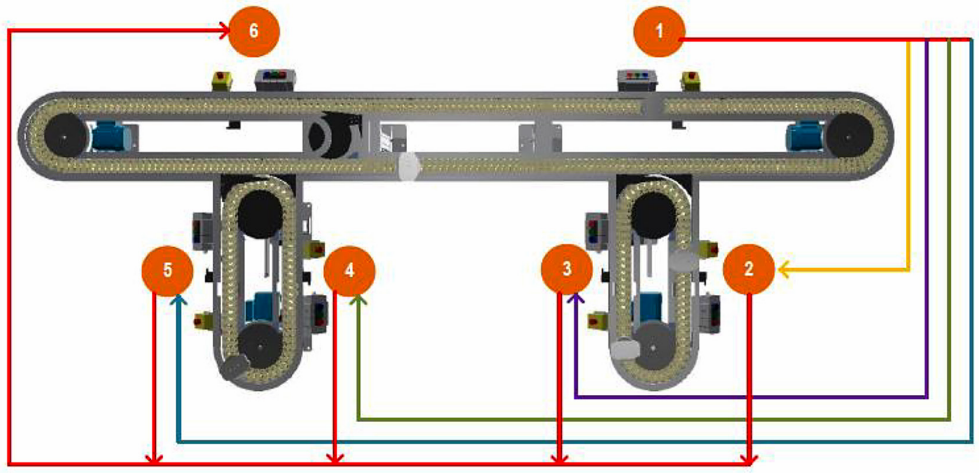

Fig. 1. Universal production station

Next, we present the CP model for the described problem. The following notation is used in the problem.

Indices and sets:
V – initial number of jobs;
q – index for initial jobs, $q = 1,...,V$;
W_q – initial number of operations of a job q;
g – index for operations of job q, $g = 1,...,W_q$;
M – number of machines;
i – index for machines, $i = 1,...,M$;
N – number of jobs;
j – index for jobs, $j = 1,...,N$;
O_j – number of operations of job j;
k,l – index for operations of job j, $k,l = 1,...,O_j$.

Parameters and indices:
r_q – number of times the job q must be repeated;
h – index for job repetitions, $h = 1,...,r_q$;
s_g – number of times the operation g must be repeated;
p_k – position of the operation k in the sequence of operations of the current job;
t_{kij} – processing time of job j on a machine i for operation k.

The number of jobs considering job repetitions:

$$N = \sum_{q=1}^{V} r_q$$

The number of operations in a job j considering operation repetitions is the same as in case with job q but the duration of the operation increases by operation repetition coefficient s_g. Here index q must be appropriately converted to j because of jobs repetitions.

$$O_q = W_q$$

The position of the last operation of the job j is O_j. Processing time of a job j on a machine considering operation repetitions is $s_k t_{kij}$.

The CP model for the problem is based on the following decision variables:
x_{kij} – an optional interval variable to express the processing time t_{kij} of operation k of the job j on the machine i;
y_{kj} – an interval variable to express the operation k of the job j;
z_i – a sequence variable having the order of the x_{kij} interval variable on the machine i.

Minimize the makespan:

$$\min \max_{j=1,...,N,k=O_j} \text{endOf}(y_{kj}) \quad (1)$$

Subject to:

A given operation l must start after the previous operation k is finished:

$$\forall(j,k,l : p_l = 1 + p_k)[\text{endBeforeStart}(y_{kj}, y_{lj})] \quad (2)$$

A given job j cannot be processed simultaneously on two or more machines:

$$\forall(k)[\text{alternative}(y_{kj}, \text{all}(s_k t_{kij}) x_{kij})] \quad (3)$$

A single job at a time can be produced on a machine i:

$$\forall(i)[\text{noOverlap}(z_i)] \quad (4)$$

HEURISTICS

Heuristic techniques are usually applied to problem-solving that uses a practical approach that is not always optimal or reasonable but is nonetheless acceptable for achieving a quick,

approximate, or immediate result. Heuristic methods are used to expedite the process of obtaining a workable solution in situations when it is impractical or unattainable to find an ideal one.

Figure 2 explains some initial steps that must be executed, providing a heuristic rule as a parameter. A set of operations that must be assigned to machines must be created, duplicating a definite number of times the operations that must be repeated in each task. Next, the appropriate order to the operations according to seven heuristic rules is applied. After that, the standard sorting order of task ID, repeated task ID, and operation ID are applied despite the selected heuristic rule.

There are some beginning operations that must be executed only on the first machine. There are also some final operations that must be executed after all operations on the last machine only. Other operations may be executed on other (medium) machines. Figure 3 explains the steps of assigning operations to machines. Operations to the first and the last machines are assigned in sequential order of operation. Assigning operations to medium machines is executed in the order set by appropriate heuristics H1-H7 which are explained in Figure 2. Generally, all operations of one task are assigned to one machine. The operations of the next task after sorting them are assigned to the next available machine. In the end, the start time and end time for each task after executing operations on machines are calculated. At this step, there is no check if the constraints on operations and machines are satisfied.

Figure 4 explains the process of checking constraint satisfaction and correction of assignment operations on machines by shifting the operations. Check if operations clash exists (operations of a task must be executed sequentially after the previous operation is finished). If yes, calculate the time that must be added to the clashed operations and shift them to machines. Also, shift operations are executed later on the given machine. Next, calculate the total execution time on each machine. Finally, calculate the start and end time for each machine. Check if a machine clash exists (the machine must execute only one operation at a time). If yes, calculate the time that must be added to operations on a clashed machine. Next, calculate the total execution time on each machine. Finally, calculate the start and end time for each machine.

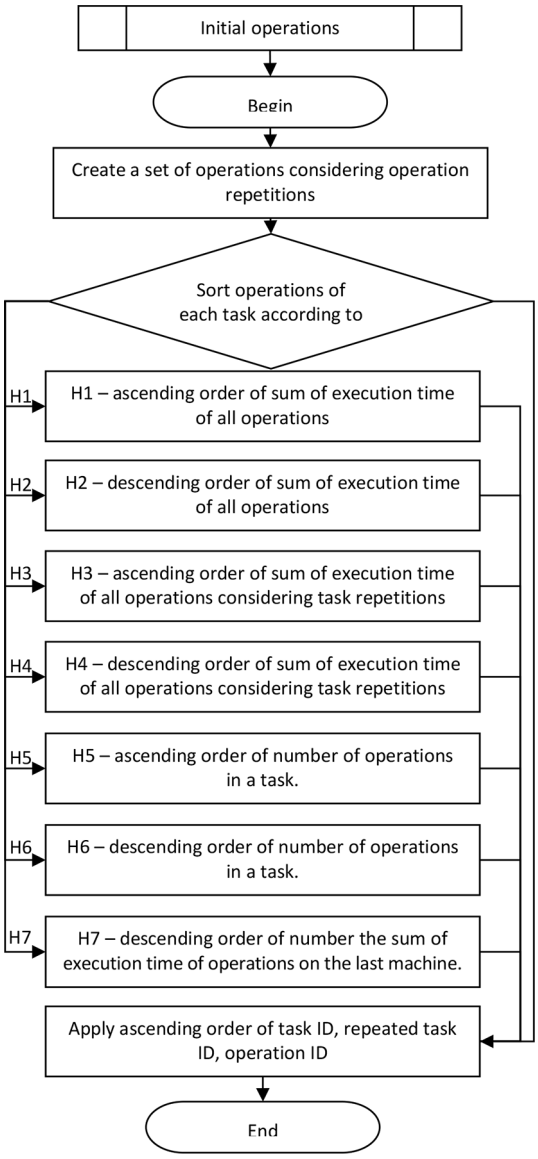

Fig. 2. Procedure "Initial operations"

Figure 5 presents the complete procedure of calculating makespan. In the beginning, initial operations are executed by applying the appropriate order of tasks and operations. Next, assigning operations to machines is performed. Lately, shifting operations on machines are iteratively executed while the constraints are not satisfied, and correction of assigning operations to machines is needed. Makespan is the sought value.

Proposed heuristics advantages:

- Fast execution.
- Simplicity of implementation.
- Producing a solution always which satisfies all constraints.
- Possibility of usage as a reference when checking the performance of other methods.

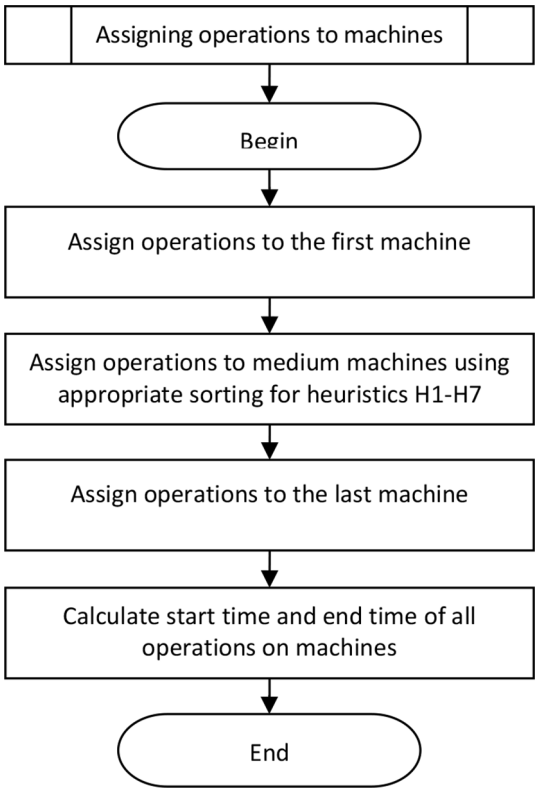

Fig. 3. Procedure "Assigning operations to machines"

Proposed heuristics disadvantages:

- Operation sorting is performed at the beginning before assigning them to machines, so later, the idle time on machines could appear.
- The quality of results achieved is rather low.

EXPERIMENT

The purpose of the computational experiment was to evaluate the solution quality found by CPO in the CPLEX solver and to compare the results with some heuristics.

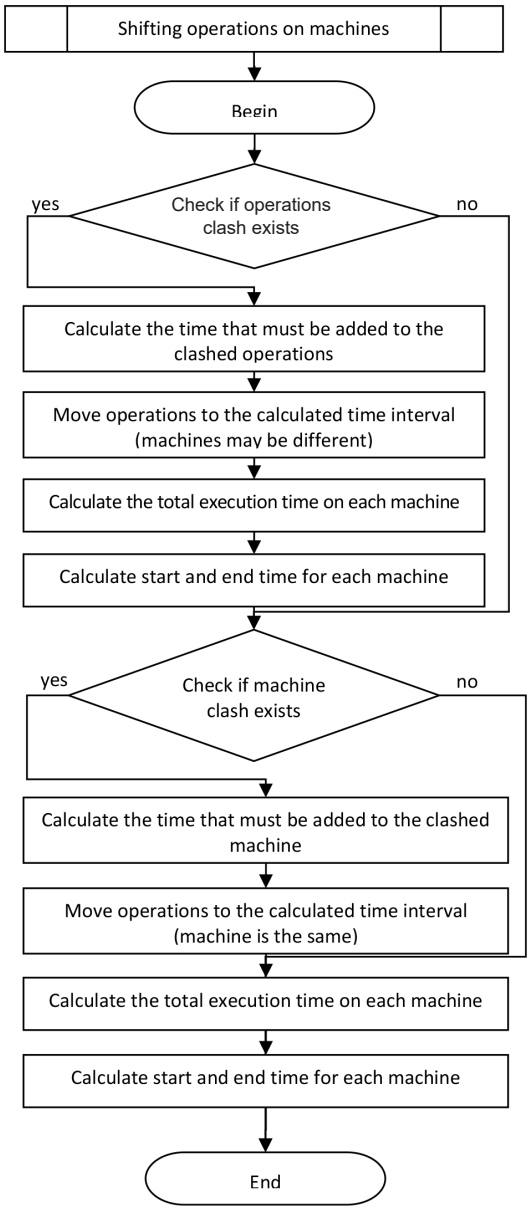

Fig. 4. Procedure "Shifting operations on machines"

The experiments were performed using the SQL Server 2019.0150.2000.05, SQL Server Management Studio 15.0.18424.0.

Table 1. Number of tasks by instances

Instance	Number of tasks	Number of tasks with task repetitions	Instance	Number of tasks	Number of tasks with task repetitions
1	4	31	11	9	30
2	4	23	12	9	53
3	5	29	13	10	63
4	5	40	14	10	45
5	6	20	15	11	49
6	6	34	16	11	51
7	7	56	17	12	50
8	7	52	18	12	81
9	8	36	19	13	70
10	8	51	20	13	79

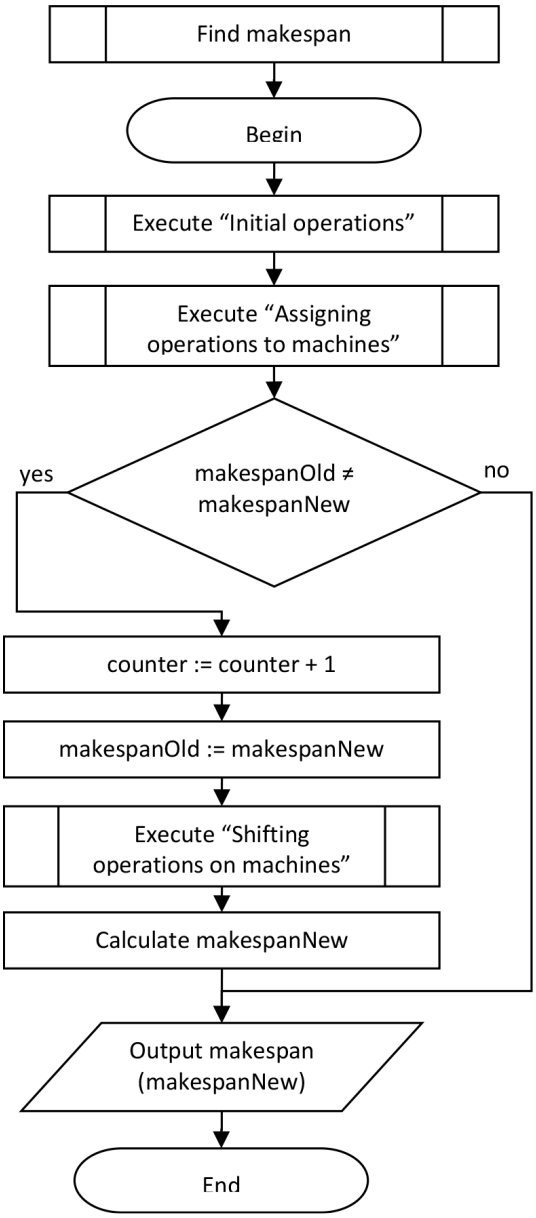

Fig. 5. Procedure "Find makespan"

The computer parameters were:

- Processor: AMD Ryzen 5 1600 Six-Core Processor 3.20 GHz
- System type: 64-bit Operation System, x64-based processor
- RAM: 16 GB
- Operation system: Windows 10

For the experiment, 20 instances were generated in which there were from 4 to 13 tasks, but from 20 to 81 tasks considering task repetitions (Table 1). In each task, there were from 4 to 12 operations, but from 9 to 43 operations considering operation repetitions (Table 2). In this instance, there were from 61 to 313 operations considering operation repetitions (Table 3). Table 4 reports the number of machines by instance. So there were from 6 to 10 machines per instance.

Table 5 reports the performance of heuristics per instance with the identification of which heuristics found the best solution among all heuristics, i.e. the solution with minimum makespan. Heuristics 1 was the best among all heuristics, which found the 9 solutions with minimum makespan from 20 instances. Heuristics 3-5 found only 1 best solution per instance. Despite this fact, other heuristics didn't find the same solution. So mentioned heuristics 3-5 as also important. For the 1st instance, two heuristics (heuristics 3 and 7) found the same solution, which was the best among all heuristics. For other instances, only one heuristic found the best solution among all. Therefore all heuristics are noteworthy.

The next experiments were performed with the use of CPO solver. It has been executed once for each instance. The results of scheduling are summarised in Table 6, which compares the performance of heuristics and CPLEX Constraint Programming Optimizer. For 10 instances, an optimal solution was identified by CPO. For the rest instances, the solution which was found was far enough from the optimal one from 0.30% up to 94.49% with an average value 75.84%, which is low enough and identifies that more efficient algorithms should be implemented.

Heuristics found the optimal solution only for one instance. On average, the heuristics solution was worse than CPO 19.37% up to 39.17%. But comparing the heuristics with the CPO optimal solutions, heuristics was worse than CPO, slightly less on average 17.68% up to 39.17%.

All heuristics found the solution in less than a second. The time limit for CPO was set to 30 seconds.

Experiments show that Constraint Programming can result in significant benefits compared to heuristics. The CP problem formulation may readily support more flexible definitions of choice variables and all extra practical restrictions due to the wide library of variables and constraints contained inside the CP tools for addressing optimization issues. Experimental evidence supports the claim that CP can produce high-quality results quickly.

Table 2. Number of operations by tasks

Instance	Task	Number of operations	Number of operations with operation repetitions	Instance	Task	Number of operations	Number of operations with operation repetitions
1	1	5	19	13	8	9	24
1	2	4	12	13	9	9	22
1	3	5	15	13	10	6	18
1	4	5	19	14	1	8	28
2	1	6	17	14	2	9	31
2	2	4	13	14	3	8	27
2	3	7	16	14	4	7	24
2	4	6	15	14	5	11	25
3	1	8	27	14	6	11	33
3	2	6	18	14	7	10	33
3	3	6	14	14	8	10	26
3	4	6	15	14	9	8	23
3	5	7	26	14	10	8	24
4	1	7	13	15	1	10	29
4	2	7	19	15	2	7	24
4	3	6	10	15	3	6	18
4	4	6	20	15	4	7	29
4	5	6	19	15	5	8	21
5	1	5	15	15	6	7	27
5	2	6	18	15	7	8	20
5	3	5	18	15	8	9	25
5	4	5	16	15	9	7	20
5	5	5	16	15	10	8	20
5	6	7	17	15	11	5	9
6	1	8	24	16	1	7	25
6	2	8	18	16	2	8	22
6	3	8	22	16	3	9	27
6	4	8	26	16	4	9	29
6	5	9	29	16	5	8	23
6	6	10	24	16	6	7	23
7	1	9	29	16	7	7	23
7	2	10	36	16	8	6	17
7	3	9	31	16	9	7	25
7	4	9	22	16	10	7	19
7	5	9	24	16	11	6	21
7	6	10	35	17	1	8	21
7	7	10	40	17	2	10	25
8	1	10	30	17	3	7	16
8	2	8	26	17	4	10	31
8	3	10	25	17	5	7	20
8	4	6	22	17	6	6	22
8	5	8	28	17	7	8	25
8	6	9	30	17	8	9	29
8	7	9	31	17	9	7	18
9	1	12	35	17	10	9	23
9	2	8	27	17	11	9	26
9	3	10	28	17	12	9	30
9	4	9	25	18	1	8	24
9	5	8	27	18	2	8	24
9	6	8	25	18	3	6	16
9	7	6	17	18	4	8	30
9	8	10	22	18	5	7	20
10	1	10	29	18	6	9	32
10	2	10	22	18	7	7	24
10	3	9	27	18	8	8	18
10	4	9	23	18	9	9	27
10	5	8	16	18	10	8	26
10	6	9	20	18	11	8	23
10	7	12	43	18	12	7	20

Table 2. Cont.

Instance	Task	Number of operations	Number of operations with operation repetitions	Instance	Task	Number of operations	Number of operations with operation repetitions
10	8	8	27	19	1	9	35
11	1	6	19	19	2	9	26
11	2	8	28	19	3	6	24
11	3	9	24	19	4	10	28
11	4	8	24	19	5	9	32
11	5	8	21	19	6	7	19
11	6	8	20	19	7	6	18
11	7	8	18	19	8	7	29
11	8	8	24	19	9	7	22
11	9	10	36	19	10	6	20
12	1	9	25	19	11	7	19
12	2	5	9	19	12	8	22
12	3	7	29	19	13	7	19
12	4	9	34	20	1	6	19
12	5	9	30	20	2	6	17
12	6	8	33	20	3	7	23
12	7	10	24	20	4	7	24
12	8	9	31	20	5	9	33
12	9	7	17	20	6	8	28
13	1	10	23	20	7	10	32
13	2	7	28	20	8	7	24
13	3	9	14	20	9	7	20
13	4	9	22	20	10	8	27
13	5	7	17	20	11	5	10
13	6	9	29	20	12	6	18
13	7	8	15	20	13	9	26

Table 3. Number of operations by instance

Instance	Number of operations with operation repetitions	Instance	Number of operations with operation repetitions
1	65	11	214
2	61	12	232
3	100	13	212
4	81	14	274
5	100	15	242
6	143	16	254
7	217	17	286
8	192	18	284
9	206	19	313
10	207	20	301

Table 4. Number of machines by instance

Instance	Machine number	Instance	Machine number
1	6	11	8
2	6	12	8
3	6	13	9
4	6	14	9
5	7	15	9
6	7	16	9
7	7	17	10
8	7	18	10
9	8	19	10
10	8	20	10

Table 5. Performance of heuristics per instance

Instance	Heuristics 1	Heuristics 2	Heuristics 3	Heuristics 4	Heuristics 5	Heuristics 6	Heuristics 7
1			√				√
2						√	
3		√					
4	√						
5							√
6		√					
7	√						
8	√						
9	√						
10	√						
11				√			
12	√						
13			√				
14	√						
15			√				
16			√				
17							√
18	√						
19	√						
20					√		
Total	9	2	4	1	1	1	3

Note: √ – the best solution among all heuristics.

Table 6. Performance of heuristics and CPLEX constraint programming optimizer

Instance	Gap from heuristics to CPLEX best objective	Gap from heuristics to CPLEX where best objective equals best bound	Gap from CPLEX best objective to best bound	Optimal solution
1	13.94%		76.46%	
2	0.00%	0.00%		√
3	36.54%	36.54%		√
4	23.76%		89.20%	
5	25.00%		70.19%	
6	26.74%		89.94%	
7	25.07%	25.07%		√
8	31.01%		90.31%	
9	20.81%		86.75%	
10	12.37%			
11	12.09%	12.09%		√
12	30.97%		84.89%	
13	18.97%		0.30%	
14	9.93%	9.93%		√
15	17.14%	17.14%		√
16	39.17%	39.17%		√
17	4.35%	4.35%		√
18	6.97%		94.49%	
19	9.34%	9.34%		√
20	23.17%	23.17%		√
Min	0.00%	0.00%	0.30%	
Avg	19.37%	17.68%	75.84%	
Max	39.17%	39.17%	94.49%	

CONCLUSIONS

Production can be planned very precisely for a small business using only basic tools; therefore, the company needs the best options from a special system. A certain margin of error is acceptable for a large factory with several dozen manufacturing lines and orders numbering in the hundreds per day because the goal is to optimize a problem that is difficult for a human to solve. The accountability of such a system is crucial for both sorts of users since the plan must be established quickly enough to avoid discouraging the use of the software.

Problems involving mathematical optimization can be solved in numerous ways. Among the methods are local search, evolutionary algorithms, constraint programming, mixed integer programming, and greedy algorithms. One approach might be more effective than the other, based on the problem size, nature, and desired quality of the solution to the problem.

One of the most challenging but crucial aspects of manufacturing is production scheduling. In this research, we show how the practical production FFSSP can be implemented using the CP. The ideas discussed here are easily adaptable to a wider range of issues. We demonstrated that CP methods can be successfully applied to scheduling problems in the production planning sector. Based on the proposed CP model, the practitioners have been able to create more complex problems.

We also proposed seven heuristics which demonstrate generally worse results than the CP. But implemented heuristics could be a part of more complex planning and problem-solving metaheuristics in future research.

CP is a paradigm for resolving combinatorial issues which incorporates numerous methods from operations research, computer science, and artificial intelligence. Programmers can declare the possible constraints on the practical situations for a given set of choice variables in constraint programming. Contrary to the known programming languages, constraints indicate the characteristics of a solution to be found rather than an algorithm to be implemented. Programmers must also indicate a way to solve the restrictions in addition to the set of constraints. CP is a crucial tool for engineering and manufacturing since it has a significant effect on how productive a process is.

Acknowledgements

The studies were realised with a support from statutory activity financed by Polish Ministry of Science and Higher Education (0613/SBAD/4770).

REFERENCES

1. Lasi, H., Fettke, P., Kemper, H.G., Feld, T. and Hoffmann, M. (2014). Industry 4.0. Business & Information Systems Engineering, 6, 239–242. DOI: https://doi.org/10.1007/s12599-014-0334-4
2. Teppan, E. (2022). Types of Flexible Job Shop Scheduling: A Constraint Programming Experiment. Proceedings of the 14th International Conference on Agents and Artificial Intelligence, Vol. 3: ICAART, 516–523. https://doi.org/10.5220/0010849900003116
3. Yin, Y., Stecke, K.E., and Li, D. (2018). The evolution of production systems from Industry 2.0 through Industry 4.0. International Journal of Production Research, 56(1–2), 848–861. DOI: https://doi.org/10.1080/00207543.2017.1403664
4. Da Col, G. and Teppan, E.C. (2019). Google vs IBM: A constraint solving challenge on the job-shop scheduling problem. Electronic Proceedings in Theoretical Computer Science, EPTCS, 306, 259–265. DOI: https://doi.org/10.4204/EPTCS.306.30
5. Delfanti, A. and Frey, B. (2021). Humanly Extended Automation or the Future of Work Seen through Amazon Patents. Science Technology and Human Values, 46(3), 655–682. DOI: https://doi.org/10.1177/0162243920943665
6. Sparrow, R. and Howard, M. (2017). When human beings are like drunk robots: Driverless vehicles, ethics, and the future of transport. Transportation Research Part C: Emerging Technologies, 80, 206–215. DOI: https://doi.org/10.1016/j.trc.2017.04.014
7. Pinedo, M.L. (2016). Scheduling: Theory, Algorithms, and Systems. NewYork: Prentice-Hall.
8. Da Col, G. and Teppan, E.C. (2022). Industrial-size job shop scheduling with constraint programming. Operations Research Perspectives, 9, 100249. DOI: https://doi.org/10.1016/j.orp.2022.100249
9. Krystek, J. and Kozik, M. (2012). Analysis of the job shop system with transport and setup times in deadlock-free operating conditions. Archives of Control Sciences, 22(4), 417–425. DOI: https://doi.org/10.2478/v10170-011-0032-0
10. Applegate, D. and Cook, W. (1991). Computational study of the job-shop scheduling problem. ORSA Journal on Computing, 3(2), 149–156. DOI: https://doi.org/10.1287/ijoc.3.2.149

11. Rossi, F., Beek, P. van, and Walsh, T. (2006). Handbook of Constraint Programming (Foundations of Artificial Intelligence). In Handbook of Constraint Programming.
12. Pinedo, M.L. (2016). Scheduling: Theory, algorithms, and systems, fifth edition. In Scheduling: Theory, Algorithms, and Systems, Fifth Edition. DOI: https://doi.org/10.1007/978-3-319-26580-3
13. Zäpfel, G. and Wasner, M. (2000). Heuristic solution concept for a generalized machine sequencing problem with an application to radiator manufacturing. International Journal of Production Economics, 68(2), 199–213. DOI: https://doi.org/10.1016/S0925-5273(98)00229-1
14. Abderrahim, M., Bekrar, A., Trentesaux, D., Aissani, N., and Bouamrane, K. (2022). Bi-local search based variable neighborhood search for job-shop scheduling problem with transport constraints. Optimization Letters, 16(1), 255–280. DOI: https://doi.org/10.1007/s11590-020-01674-0
15. Respen, J., Zufferey, N., and Amaldi, E. (2016). Metaheuristics for a job scheduling problem with smoothing costs relevant for the car industry. Networks, 67(3), 246–261. DOI: https://doi.org/10.1002/net.21656
16. Homayouni, S.M. and Fontes, D.B.M.M. (2021). Production and transport scheduling in flexible job shop manufacturing systems. Journal of Global Optimization, 79(2), 463–502. DOI: https://doi.org/10.1007/s10898-021-00992-6
17. Beauchemin, M., Ménard, M.-A., Gaudreault, J., Lehoux, N., Agnard, S., and Quimper, C.-G. (2022). Dynamic allocation of human resources: case study in the metal 4.0 manufacturing industry. International Journal of Production Research. DOI: https://doi.org/10.1080/00207543.2022.2139002
18. Moslehi, G. and Mahnam, M. (2011). A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. International Journal of Production Economics, 129(1), 14–22. DOI: https://doi.org/10.1016/j.ijpe.2010.08.004
19. Bautista-Valhondo, J. and Alfaro-Pozo, R. (2020). Mixed integer linear programming models for Flow Shop Scheduling with a demand plan of job types. Central European Journal of Operations Research, Vol. 28, No. 1. DOI: https://doi.org/10.1007/s10100-018-0553-8
20. Jolai, F. and Abedinnia, H. (2013). Consideration of transportation lags in a two-machine flow shop scheduling problem. Scientia Iranica, 20(6), 2215–2223.
21. Quadt, D. and Kuhn, H. (2007). A taxonomy of flexible flow line scheduling procedures. European Journal of Operational Research, 178(3), 686–698. DOI: https://doi.org/10.1016/j.ejor.2006.01.042
22. Ruiz, R. and Vázquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. European Journal of Operational Research, 205(1), 1–18. DOI: https://doi.org/10.1016/j.ejor.2009.09.024
23. Hasani, A., Hosseini, S.M.H., and Sana, S.S. (2022). Scheduling in a flexible flow shop with unrelated parallel machines and machine-dependent process stages: Trade-off between Makespan and production costs. Sustainability Analytics and Modeling, 2, 100010. DOI: https://doi.org/10.1016/j.samod.2022.100010
24. Ribas, I., Leisten, R., and Framinan, J.M. (2010). Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. Computers and Operations Research, 37(8), 1439–1454). DOI: https://doi.org/10.1016/j.cor.2009.11.001
25. Eskandari, H. and Hosseinzadeh, A. (2014). A variable neighbourhood search for hybrid flow-shop scheduling problem with rework and set-up times. Journal of the Operational Research Society, 65(8), 1221–1231. DOI: https://doi.org/10.1057/jors.2013.70
26. Luo, H., Zhang, A., and Huang, G.Q. (2015). Active scheduling for hybrid flowshop with family setup time and inconsistent family formation. Journal of Intelligent Manufacturing, 26(1), 169–187. DOI: https://doi.org/10.1007/s10845-013-0771-9
27. De Abreu, L.R., Araújo, K.A.G., de Athayde Prata, B., Nagano, M.S., and Moccellin, J.V. (2022). A new variable neighbourhood search with a constraint programming search strategy for the open shop scheduling problem with operation repetitions. Engineering Optimization, 54(9), 1563–1582. DOI: https://doi.org/10.1080/0305215X.2021.1957101
28. Roundy, R., Chen, D., Chen, P., Çakanyildirim, M., Freimer, M.B., and Melkonian, V. (2005). Capacity-driven acceptance of customer orders for a multi-stage batch manufacturing system: Models and algorithms. IIE Transactions (Institute of Industrial Engineers), 37(12), 1093–1105. DOI: https://doi.org/10.1080/07408170500288042
29. Kopanos, G.M., Méndez, C.A., and Puigjaner, L. (2010). MIP-based decomposition strategies for large-scale scheduling problems in multiproduct multistage batch plants: A benchmark scheduling problem of the pharmaceutical industry. European Journal of Operational Research, 207(2), 644–655. DOI: https://doi.org/10.1016/j.ejor.2010.06.002