

MATHEMATICAL MODELING FOR THE JOB SHOP SCHEDULING PROBLEM WITH REALISTIC CONSTRAINTS

Arthur Santana^{1,2}, Daniel Figueiredo¹, Laura Bahiense¹

¹Programa de Pós Graduação em Engenharia de Sistemas e Computação - COPPE-UFRJ
Centro de Tecnologia, Bloco H - Ilha do Fundão - Rio de Janeiro - RJ, 21941-972, Brasil

²Laboratório de Simulação e Otimização de Sistemas Produtivos - INT-MCTI

Av. Venezuela, 82 - Saúde - Rio de Janeiro - RJ, 20081-312, Brasil

{arthurss, daniel, laura}@cos.ufrj.br

RESUMO

O problema do job shop flexível determina a programação da fabricação, considerando diferentes tarefas e máquinas. Devido à generalidade e complexidade de diferentes linhas e processos de produção, os modelos para esse problema costumam ser relativamente específicos e restritos. Este trabalho considera o problema de job shop flexível com restrições mais realistas, como horários de início mínimos e diferentes funções objetivo. Dois modelos de Programação Inteira Mista (PIM) são propostos para capturar tais cenários, e são resolvidos pelo método de branch-and-bound clássico. Um software de simulação a eventos discretos, que lida com vários tipos de problemas de programação de produção e possui um núcleo heurístico, é usado para comparar a qualidade das soluções obtidas. São avaliadas instâncias de referência e outras baseadas em casos do mundo real. Os experimentos numéricos indicam que as soluções dos modelos de PIM propostos superam as soluções heurísticas em todos os casos testados.

PALAVRAS CHAVE. Problema de job shop flexível, Programação da produção, Programação inteira mista.

Tópicos do artigo: Pesquisa Operacional na Administração e Gestão da Produção. Programação Matemática.

ABSTRACT

The flexible job shop problem determines the manufacturing schedule considering different tasks and machines. Due to the generality and complexity of different production lines and processes, models for this problem are often relatively specific and restricted. This work considers the flexible job shop problem under more realistic constraints, such as minimum start times and different objective functions. Two Mixed Integer Programming (MIP) models are proposed to capture such scenarios, and they are solved by the classic branch-and-bound method. A discrete event simulation software, tackling various types of production scheduling problems and having a heuristic core, is used to compare the quality of the obtained solutions. Benchmark and real-world based instances are evaluated. Numerical experiments indicate that the solutions of the proposed MIP models outperform the heuristic solutions in all tested cases.

KEYWORDS. Flexible job shop problem, Production scheduling, Mixed-integer programming.

Paper topics: Operations Research in Production Management and Administration. Mathematical Programming.

1. Introduction

A crucial task in manufacturing operations is short-term production planning and scheduling. Ensuring that production goals are met within a relatively short deadline entails coordination and optimization of production processes. A production schedule determines what jobs are to be executed in what machines and when while satisfying a diverse set of constraints. Planning requires tracking inventory levels and modifying production schedules in response to shifts in demand, supply, or other production limitations such as machine failure. Thus, planning and scheduling are often jointly addressed in manufacturing operations.

To be successful, companies must properly plan and manage their scarce resources during manufacturing. A good plan and schedule can guarantee that the desired amount and quality is produced at the appropriate time, while also reducing waste, downtime, and other costs. Thus, constructing effective plans and schedules is fundamental for modern manufacturing, and this gives rise to the Job Shop Scheduling Problem (JSSP). Indeed, there is a large body of literature in tackling these problems using a broad variety of problem formulation [Xie et al., 2019].

Many real-world manufacturing problems are also incredibly complex and very specific in terms of constraints and actions. Nevertheless, recent academic works have addressed planning and scheduling problems that increasingly resemble real-world scenarios [Türkyılmaz et al., 2020]. A typical strategy is to optimize the production plan using mathematical programming techniques such as Mixed-Integer Programming (MIP). The capacity of examining alternative production schedules and selecting the most efficient schedule is a key benefit of employing MIP in JSSP.

Manufacturers can use mathematical models to plan production efficiently (or optimally) and allocate resources by considering production constraints like finite resources or machine capacity in complex production environments [Gomes et al., 2005]. Furthermore, such models can be updated during production to reflect changes in the manufacturing environment, allowing manufacturers to adjust their schedules in real-time to account for new constraints or shifting customer demand. Thus, MIP is an indispensable tool for modern manufacturing processes.

The main objective of this work is to propose mathematical models that address the constraints and objectives of a real-world manufacturing environment. To achieve this, the first contribution of this paper is to identify the essential features for the models to better capture real-world production problems. A second contribution is the proposal of mathematical models that cover all the identified constraints. A third contribution is the evaluation of benchmark instances and instances derived from real-world manufacturing problems using these models. Results indicate that the proposed mathematical models yield optimal solutions for medium-sized benchmark instances. On simplified real-world instances, the solutions obtained outperform the solutions provided by a state-of-the-art tool used in manufacturing processes.

The remainder of this paper is organized as follows. Section 2 provides a succinct review of the relevant literature. Section 3 describes real-world production constraints and illustrates the issues addressed by this work. The proposed mathematical models are presented in Section 4. Numerical results on benchmark and real-world instances are presented in Section 5. Finally, Section 6 concludes the paper.

2. Related work

Mathematical models that represent the increasingly complex reality of factories are still much needed, and for this reason, studies have been carried out with the objective of improving the performance and comprehensiveness of the models over the years.

A mathematical modeling and a hybrid metaheuristic (hybridization of Artificial Immune and Simulated Annealing Algorithms) for Flexible JSSP (F-JSSP) is introduced by [Roshanaei et al.,

2013]. Although they have addressed the basic constraints (e.g. Sequence Dependent Setup Times (SDST)) of the F-JSSP with excellence, bringing a structurally efficient formulation, they do not take into account real constraints of this context such as deadlines, startup constraints and, more importantly, they do not address problems where a machine needs to be set up between two operations of the same job.

The survey provided by [Chaudhry e Khan, 2016] covers techniques that have been used to tackle F-JSSP. Among the techniques used in the reviewed papers are metaheuristics such as Ant-Colony Optimization (ACO), Evolutionary Algorithms (EAs), heuristics, exact algorithms such as MIP, and hybrid algorithms using one or more heuristics and/or metaheuristics.

Though simple in comparison to the actual complexity present in today's factories, [Pinedo, 2016] contains numerous very well-presented concepts, mathematical formulations and heuristics that serve as the foundation for numerous studies of various techniques that are developed to deal with the F-JSSP.

A Mixed Integer Linear Programming (MILP) formulation is presented by [Shen et al., 2018] to solve F-JSSP with SDST. To further improve the solution quality and efficiency, the article proposes a tabu search algorithm with specific neighborhood functions and a diversification structure, which begins with a randomly generated initial solution, then applies feasibility checks and iterative searches for better solutions by employing local search procedures and making minor changes to the current solution. They also emphasize the importance of considering setup times in manufacturing line problems because setup times can have a significant impact on overall manufacturing efficiency and productivity.

An integer non-linear model is introduced by [Winklehner e Hauder, 2022] to deal with F-JSSP-SDST, alongside by a real-world case of a manufacturer in the automotive sector producing injection molded parts. What makes this work interesting is that it takes into account other factors such as release dates and deadlines. By including release dates and deadlines in the mathematical model, the proposed approach can better reflect the constraints and objectives of a practical manufacturing environment.

A Mixed Integer Non-Linear Programming (MINLP) model for F-JSSP with variable batches is proposed by [Xiuli et al., 2021], that also takes into account a transfer time between operations processed on different machines. In their work, jobs can be divided into sub-batches so that they can be processed on different machines at the same time. This is a significant improvement over traditional F-JSSP models, in which each operation of a job must be completed on a single machine before proceeding to the next.

All the related works presented have some aspects of real-world problems identified in this paper, although there is a gap, which this paper aims to fill, when it comes to dealing with all aspects together.

3. Real-world manufacturing constraints

A mathematical model is considered efficient and effectively used for scheduling in production lines, when its constraints must reflect the characteristics of the real world whose model it is intended to represent [Chaudhry e Khan, 2016]. In this section the relevant characteristics commonly found in the context of F-JSSP production lines are presented.

One of the main characteristics of F-JSSP is the need to set up a machine to start producing an item after it has been used to produce another kind of item. Often this set up time depends on both kinds of items: produced before and to be produced [Shen et al., 2018], and is called sequence dependent setup time. It is well known that setup times have a major impact on scheduling performance [Kim e Bobrowski, 1994]. This effect stems from the need for machine adjustments for

different reasons, such as the difference in dimensions of the items produced, different production temperatures (which implies heating or cooling of the machine), or even the change of large tools to be used (e.g. molds).

Another set of constraints commonly present in real production lines are related to dates [Yeung et al., 2010]. The start of production is sometimes limited to the arrival of raw materials, or it may even depend on a process that is performed in another part of the factory. This means that the demand exists, but it should not be started before a predetermined time. In the latter situation, this type of constraint can be avoided if the planning scope is expanded to cover all sectors responsible for one or more operations on a certain item [Pinha et al., 2015]. However, it is not unusual to find contexts so complex that, to make the planning process feasible, it becomes necessary to break the planning of the whole plant into smaller parts, such as physical sectors or related areas.

On the other hand, deadlines for the production of a demand are also part of the planning routine in real factories. In production lines that operate both by pushed or pulled demand (which generally involves a strict inventory control), deadlines are necessary to control what to perform at each moment of planning. The difference is that in pushed production these deadlines tend to be more flexible, because the demand deadlines are intended to ensure the minimum safety levels of operations. While in pushed production, this deadline can be crucial to the organization's success, especially in cases where deadlines are set through agreements with end customers. In this case, it is clear that meeting or not meeting a deadline has immediate effects on customer relationship management (while also accruing early bonuses or paying late fees).

To be used in real production environments, a software that proposes to generate viable solutions to this problem must be adherent to the reality it intends to mimic. The following subsection presents, very briefly, a software that, based on the concept of digital twins, encompasses the most diverse characteristics present in the production environment.

3.1. Discrete event simulation heuristics

As a basis for comparison with the solutions generated by the mathematical model here proposed, a state-of-the-art software used by manufacturing companies that generates solutions for scheduling problems was used for comparison. The software uses a processing kernel called See The Future (STF), developed by the National Institute of Technology (INT), and is made available through a customized layer for each client.

The STF kernel generates viable solutions for a large variety of scheduling problems using different heuristics and user guidance. The most common approach is to build a viable solution through a greedy and iterative procedure that considers at each step all possible combinations of jobs, operations, and machines (a triplet) among the jobs and operations yet to be allocated.

To compare different allocations, different queueing policies are used in descending order of priority: First In, First Out (FIFO); Shortest Deadline (SD); Shortest Setup Time (SST); and Shortest Processing Time (SPT). Whenever two triples share the same resources within the planning horizon, these queueing rules take effect. The triplet that wins across all comparisons in a given iteration becomes a processing task in the viable solution (and is not reconsidered).

Note that STF does not have an explicit objective function. However, the user can determine different kinds of priorities (e.g., reduce tardiness, reduce idleness) that are considered when assessing triplets to generate a viable schedule.

4. Mathematical model

The first model proposed in this work to solve more realistic instances of the F-JSSP is a MILP that has both classic constraints, commonly found in mathematical modeling papers on F-JSSP, and additional constraints, that better represent real production environments. While these

constraints are not new, they are usually found separately, in different works, and under different modeling perspectives. Thus, the proposed model consolidates prior ideas into a single and more realistic mathematical model.

The proposed model assumes that start times, processing times, and setup times are integers, assuming that time intervals smaller than seconds are negligible, and therefore time intervals can be made discrete and measured in minutes or hours. Table 1 describes the models' indices and parameters, and Table 2 describes the decision variables.

Table 1: Indexes and parameters.

Notation	Description
n	Number of jobs.
m	Number of machines.
i	Index for machines, where $1 \leq i \leq m$.
j, h	Indices for jobs, where $1 \leq j, h \leq n$.
n_j, n_h	Number of operations of jobs j and h .
l, k	Indices for operations of a job, where $1 \leq l \leq n_j, 1 \leq k \leq n_h$.
M	A large positive number.
$R_{j,l}$	The set indicating the machines eligible to process operation l of job j .
$P_{i,j,l}$	Processing time of l th operation of job j on machine i .
$O_{i,j,l,h,k}$	Sequence dependent setup times between l th operation of job j that precedes k th operation of job h on machine i .
Q_j	Minimum starting time for job j .
D_j	Deadline for job j .

Table 2: Decision variables.

Notation	Description
$y_{i,j,l}$	Binary decision variable indicating whether job j is processed in machine i in stage l .
$x_{j,l,h,k}$	Binary decision variable indicating whether l th operation of job j is processed after k th operation of job h in any machine.
$s_{i,j,l}$	Continuous variable for starting time of operation l of job j on machine i .
b_{i,j,n_j}	Auxiliary variable indicating whether the completion of the last operation (n_j) a job j on machine i is greater than the deadline of job j .

The proposed MILP model is presented as follows:

$$\text{Minimize } \sum_{i=1}^m \sum_{j=1}^n (s_{i,j,n_j} + P_{i,j,n_j} y_{i,j,n_j}) \quad (1)$$

Subject to:

$$\sum_{i=1}^m y_{i,j,l} = 1 \quad \forall j, \forall l \quad (2)$$

$$s_{i,j,l} \leq M y_{i,j,l} \quad \forall i, \forall j, \forall l \quad (3)$$

$$\sum_{i=1}^m s_{i,j,l+1} \geq \sum_{i=1}^m s_{i,j,l} + \sum_{i=1}^m P_{i,j,l} y_{i,j,l} \quad \forall j, \forall l < n_j \quad (4)$$

$$s_{i,j,l+1} \geq s_{i,j,l} + P_{i,j,l} y_{i,j,l} + O_{i,j,l,j,l+1} - M(2 - y_{i,j,l+1} - y_{i,j,l}) \quad \forall i \in \{R_{j,l} \cap R_{j,l+1}\}, \forall j, \forall l < n_j \quad (5)$$

$$s_{i,j,l} \geq s_{i,h,k} + P_{i,h,k} + O_{i,h,k,j,l} - M(3 - x_{j,l,h,k} - y_{i,j,l} - y_{i,h,k}) \quad \forall i \in \{R_{j,l} \cap R_{h,k}\}, \forall j < n, \forall l, \forall h > j, \forall k \quad (6)$$

$$s_{i,h,k} \geq s_{i,j,l} + P_{i,j,l} + O_{i,j,l,h,k} - M(x_{j,l,h,k} + 2 - y_{i,j,l} - y_{i,h,k}) \quad \forall i \in \{R_{j,l} \cap R_{h,k}\}, \forall j < n, \forall l, \forall h > j, \forall k \quad (7)$$

$$s_{i,j,1} \geq Q_j y_{i,j,1} \quad \forall i, \forall j \quad (8)$$

$$y_{i,j,l} \in \{0, 1\} \quad \forall i, \forall j, \forall l \quad (9)$$

$$x_{j,l,h,k} \in \{0, 1\} \quad \forall j < n, \forall l, \forall h > j, \forall k \quad (10)$$

$$s_{i,j,l} \geq 0 \quad \forall i, \forall j, \forall l \quad (11)$$

The Objective Function (1) aims to minimize the makespan, and it is defined by the sum of the finishing time of the last operation of all jobs. Thus, it might be possible to reduce the sum by decreasing the finish time of each job.

Constraints (2) guarantee that an operation l of a job j can only be processed by one machine. Constraints (3) couple variables s and y , ensuring that an operation l of job j starts on machine i only if job j is processed in machine i in stage l . Thus, if $y_{i,j,l} = 0$ then $s_{i,j,l} = 0$.

Constraints (4) relates the start times of subsequent operations of the same job, establishing that a successor operation $l + 1$ of job j on machine i can only begin when its predecessor operation l on the same machine has started and finished. Constraints (5) work similarly, ensuring that the setup times of subsequent operations of the same job are respected. In fact, whenever $y_{i,j,l} = 1$ and $y_{i,j,l+1} = 1$, then the right-hand-sides of (5) are tight, and (5) are active.

Constraints (6) and (7) act together, and deal with the precedence of two operations (l and k) from different jobs (j and h), that are performed on the same machine. Necessarily, one of the operations must occur before the other, since it is not allowed for a machine to process two operations at the same time. Therefore, for these constraints to come into play, at least $y_{i,j,l}$ and $y_{i,h,k}$ must be equal to 1. Thus, whenever the l th operation of job j is processed after k th operation of job h in some machine ($x_{j,l,h,k} = 1$), then the right-hand-sides of (6) are tight, ensuring that the precedence between the operations is respected. When $x_{j,l,h,k} = 0$, Constraints (7) guarantee the same effect, but in the reverse order of precedence. This way, the setup time between two

subsequent operations is respected, since this time is added as well as the processing time for that operation on that particular machine.

Constraints (8) couple variables s and y with parameter Q , ensuring that the first operation ($l = 1$) of job j assigned to machine i starts at a minimum time Q_j . Thus, $s_{i,j,1} \geq Q_j$ whenever $y_{i,j,1} = 1$. Lastly, Constraints (9), (10) and (11) determine the decision variables' domains.

Another common objective in production line sequencing is meeting deadlines or minimizing tardiness. This alternative MINLP model is represented as follows:

$$\text{Minimize } \sum_{i=1}^m \sum_{j=1}^n (s_{i,j,n_j} + P_{i,j,n_j} - D_j) b_{i,j,n_j} \quad (12)$$

Subject to (3)-(11) and:

$$s_{i,j,n_j} + P_{i,j,n_j} y_{i,j,n_j} \geq D_j - M(1 - b_{i,j,n_j}) \quad \forall i, \forall j \quad (13)$$

$$s_{i,j,n_j} + P_{i,j,n_j} y_{i,j,n_j} \leq D_j + M b_{i,j,n_j} \quad \forall i, \forall j \quad (14)$$

$$b_{i,j,n_j} \in \{0, 1\} \quad \forall i, \forall j \quad (15)$$

Remembering that auxiliary variables b_{i,j,n_j} indicate whether the completion of a job is greater than its deadline, Constraints (13) and (14) act together, capturing the occurrence of a delay in some job j . Whenever $s_{i,j,n_j} + P_{i,j,n_j} y_{i,j,n_j} < D_j$, then $b_{i,j,n_j} = 0$; otherwise Constraints (13) would imply $s_{i,j,n_j} + P_{i,j,n_j} y_{i,j,n_j} \geq D_j$, a contradiction. Therefore, $b_{i,j,n_j} = 0$, Constraints (14) are active (tight), Constraints (13) are non-active, and no delay is accounted for in the OF. Conversely, whenever $s_{i,j,n_j} + P_{i,j,n_j} y_{i,j,n_j} > D_j$, then $b_{i,j,n_j} = 1$; otherwise Constraints (14) would imply $s_{i,j,n_j} + P_{i,j,n_j} y_{i,j,n_j} \leq D_j$, a contradiction. Therefore, $b_{i,j,n_j} = 1$, Constraints (13) are active (tight), Constraints (14) are non-active, and the corresponding delay is accounted for in the OF. In the special case $s_{i,j,n_j} + P_{i,j,n_j} y_{i,j,n_j} = D_j$, we could have either $b_{i,j,n_j} = 0$ or $b_{i,j,n_j} = 1$. As the objective function is to minimize, the solver always choose $b_{i,j,n_j} = 0$, and no delay is accounted for in the OF. Finally, Constraints (15) determine the domain of b_{i,j,n_j} variables. It is worth mentioning that (12) is the sum of the difference between the finish time of a job and its deadline across all jobs. Therefore, it might be possible to reduce the value of this function by delaying a job aiming to finish others earlier.

5. Experimental results and discussion

This section presents the performance of both proposed mathematical models and a direct comparison with the STF heuristic. A set of 20 benchmark instances named Fattahi Setup [Fattahi et al., 2007] were modified by adding start time and deadline constraints to contain all characteristics captured by the proposed models. We call these instances Fattahi Setup Modified (FSM). A set of three instances based on real-world problems provided by INT and solved by STF were also considered: a plastic injection molding plant (PlastInject), a metalworking production plant (MetalMeca), and a small customized manufacturing plant (FTQL). These instances were simplified in terms of their constraints to fit the proposed models, while keeping their number of jobs, operations and machines.

The experiments were run on AMD Ryzen 7 3700x 8-Core @3.60 GHz processor with 32GB RAM. To solve the models, Python language version 3.11.3 and Gurobi solver version 10.0.1 through the API provided by the gurobipy library were used. A time limit of 3 hours was imposed

for the solver to find the optimal solution, returning the best solution when this limit is reached. The instances used and experiments implementation details are available at <https://anonymous.4open.science/r/F068/>.

Table 3 shows characteristics of all instances and corresponding results. Important attributes of the instances are the total number of jobs (n), total number of operations ($\sum_j n_j$), and total number of machines (m). The density relating the number of binary variables (# Bin) with the total number of variables (# Var) is also shown. The objective function value (OF) and the time required by the solver is presented, in seconds (s), for the MILP and MINLP models and STF. Note that STF generates a single solution for each instance and does not depend on the objective function (thus, a single execution time). The optimality gap (Opt Gap) for the models is null whenever an optimal solution is found; otherwise, a non-negative gap means that the time limit of 3 hours was reached. The STF Heuristic Gap (Heur Gap) reflects the gap with respect to the OF value returned by the respective (Makespan or Deadline) MILP or MINLP model, being calculated by the mathematical formulas $OF_{STF} - OF_{MILP}/OF_{MILP}$ and $OF_{STF} - OF_{MINLP}/OF_{MINLP}$. Therefore, to compare the performance of MILP and MINLP models against STF, we should not compare the respective Gap columns directly, but column 6 with column 12 and column 9 with column 14, respectively.

Table 3: Computational results.

Instances					MILP and MINLP Models Solutions						STF Heuristic Solutions				Time (s)
					Makespan			Deadline			Makespan		Deadline		
Instance name	# Bin / # Var	n	$\sum n_j$	m	OF (min)	Time (s)	Opt Gap	OF (min)	Time (s)	Opt Gap	OF (min)	Heur Gap	OF (min)	Heur Gap	
FSM_01	10 / 18	2	4	2	142	0.01	0.0%	80	0.01	0.0%	193	35.9%	131	63.8%	
FSM_02	7 / 13	2	4	2	204	0.01	0.0%	132	0.00	0.0%	204	0.0%	164	24.2%	
FSM_03	17 / 27	3	6	2	600	0.01	0.0%	445	0.01	0.0%	612	2.0%	496	11.5%	
FSM_04	18 / 28	3	6	2	874	0.02	0.0%	653	0.01	0.0%	1001	14.5%	878	34.5%	
FSM_05	22 / 34	3	6	2	303	0.04	0.0%	221	0.03	0.0%	321	5.9%	298	34.8%	
FSM_06	96 / 112	3	9	3	808	0.02	0.0%	626	0.03	0.0%	1216	50.5%	1085	73.3%	
FSM_07	99 / 118	3	9	5	923	0.01	0.0%	696	0.01	0.0%	952	3.1%	921	32.3%	
FSM_08	99 / 118	3	9	4	655	0.04	0.0%	519	0.06	0.0%	773	18.0%	813	56.6%	
FSM_09	99 / 118	3	9	3	553	0.07	0.0%	443	0.07	0.0%	579	4.7%	566	27.8%	
FSM_10	38 / 58	4	12	5	1899	0.05	0.0%	1599	0.04	0.0%	2000	5.3%	1883	17.8%	
FSM_11	81 / 114	5	15	6	2043	0.37	0.0%	1703	0.41	0.0%	2339	14.5%	2402	41.0%	
FSM_12	102 / 141	5	15	7	1977	0.63	0.0%	1620	0.61	0.0%	2165	9.5%	2333	44.0%	
FSM_13	149 / 197	6	18	7	2517	2.95	0.0%	2078	8.85	0.0%	3259	29.5%	3474	67.2%	
FSM_14	191 / 247	7	21	7	3317	15.92	0.0%	2741	45.86	0.0%	3699	11.5%	3939	43.7%	
FSM_15	196 / 251	7	21	7	3283	87.52	0.0%	2714	36.63	0.0%	3807	16.0%	4062	49.7%	
FSM_16	244 / 306	8	24	7	4244	3469.86	0.0%	3600	763.61	0.0%	4949	16.6%	5338	48.3%	
FSM_17	394 / 472	8	32	7	5953	10800.28	15.1%	5290	10800.43	18.4%	7268	22.1%	7383	39.6%	
FSM_18	441 / 527	9	36	8	6720	10801.38	16.6%	6088	10801.97	21.5%	8237	22.6%	8271	35.9%	
FSM_19	619 / 722	11	44	8	9709	10802.29	30.7%	8785	10802.16	33.7%	10911	12.4%	10668	21.4%	
FSM_20	735 / 847	12	48	8	11597	10801.82	32.6%	10256	10801.34	32.5%	13005	12.1%	12877	25.6%	
FTQL	47 / 71	5	24	5	70820	0.07	0.0%	34533	0.12	0.0%	88487	24.9%	52200	51.2%	
PlastInject	398 / 549	69	69	35	81298	101.66	0.0%	70636	10801.38	1.4%	89405	10.0%	123417	74.7%	
MetalMeca	4791 / 5334	61	295	21	645591	10813.41	2.8%	0	3.29	0.0%	711671	10.2%	440	***	

*** In this case, Heur Gap cannot be calculated, since there is no division by zero.

Table 3 shows that optimal solutions were found for 16/20 (80%) benchmark instances and 2/3 (67%) real-world instances for both MILP and MINLP models. For example, from instance FSM_17 to FSM_20 both the MILP and MINLP solvers were not able to find the optimal solution to within the time limit provided. For these instances, the optimality gap provided ranges from 15% to 33%. Surprisingly, optimal or very small gap solutions were found for both the very large real-world instances MetalMeca and PlastInject. It is also important to emphasize that for these instances, Heur Gaps were very large. In particular, this behavior occurred for almost all tested

instances, i.e., Heur Gap was much larger than MILP Gap or MINLP Gap, for 22/23 (95,6%) and 23/23 (100%) of the tested instances, respectively. This indicates that scheduling under the solutions of the proposed models manages resources more efficiently, allowing for increased productive capacity without increasing the number working hours, for example. It is also worth noting that large real-world instances could be easier to solve by the MILP and MINLP models than the modified benchmark instances, due to actual constraints and parameters of real-world problems.

Figures 1 and 2 illustrate the schedule for the FSM_16 instance generated by solving the MILP proposed model (makespan OF) and by STF, respectively. Despite the two schedules are similar, note that STF requires more time to finish all jobs and its gap is 16.6% with respect to the optimal solution found by the MILP model.

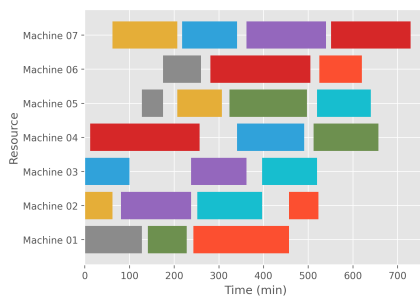


Figure 1: MILP Model Solution (makespan) for instance FMS_16.

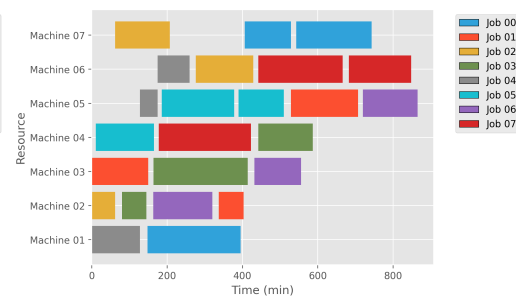


Figure 2: STF Heuristic Solution for instance FMS_16.

Similarly, Figures 3 and 4 illustrate the schedule for the FSM_17 instance generated by solving the MILP proposed model (makespan OF) and by STF. In contrast, note that the model required more time to finish all jobs (job 07 is the last one). However, by delaying the execution of job 07 the model finished many other jobs much earlier than STF, such as job 00 and job 06. Indeed, the heuristic gap considering the OF value is 22.1% with respect to the solution returned by the model. This kind of solution seems quite hard to obtain using heuristics.

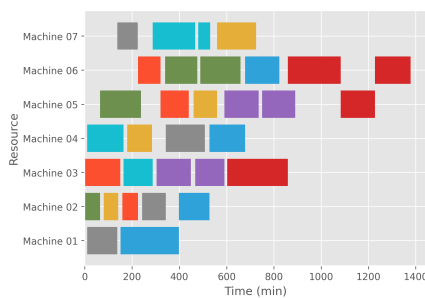


Figure 3: MILP Model Solution (makespan) for instance FMS_17.

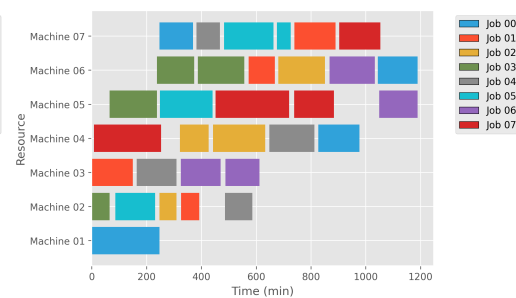


Figure 4: STF Heuristic Solution for instance FMS_17.

Figures 5 and 6 illustrate the schedule for the FSM_16 and FSM_17 instances generated by the MINLP model (deadline OF). Note that for FSM_16 the schedule generated is identical to the one generated by the MILP model (makespan OF), illustrated in Figure 1. In contrast, the schedule for FSM_17 is much tighter than the corresponding schedule generated by the makespan OF (see Figure 3). Indeed, the deadline OF forces jobs to finish closer to their deadlines, as opposed to

simply reducing their finish times.

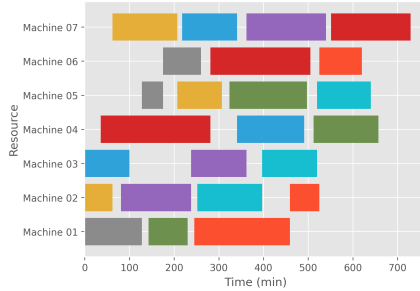


Figure 5: MINLP Model Solution (deadline) for instance FMS_16.

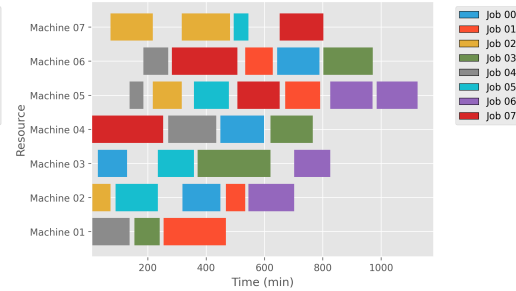


Figure 6: MINLP Model Solution (deadline) for instance FMS_17.

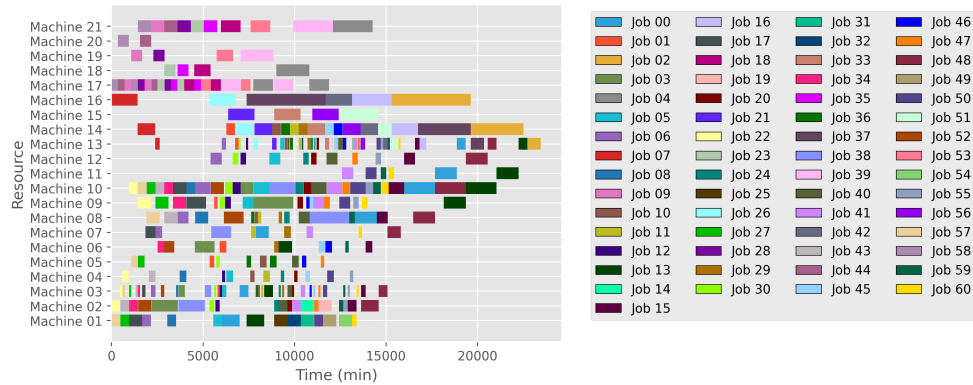


Figure 7: MILP Model Solution (makespan) for instance MetalMeca.

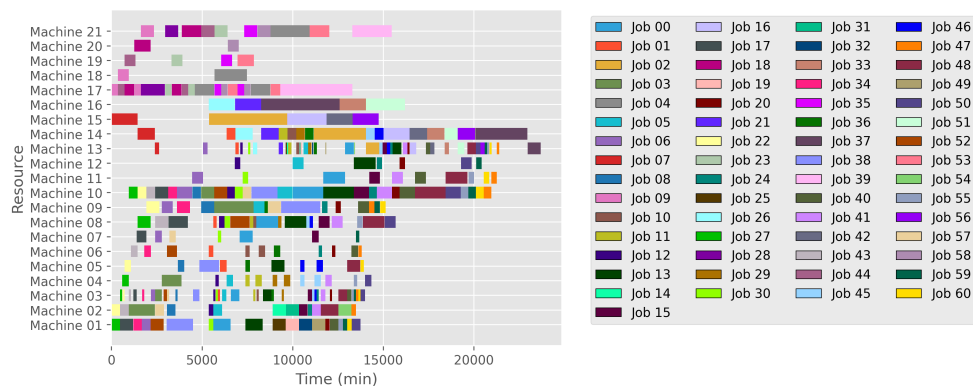


Figure 8: STF Heuristic Solution for instance MetalMeca.

Figures 7 and 8 illustrate the schedule for the MetalMeca instance (with 61 jobs) generated by solving the proposed MILP model (makespan OF) and by STF, respectively. While the two schedules look relatively similar and the heuristic gap is 10.2%, note that the schedule generated by the model has fewer jobs finishing after time 15000 in comparison to STF, where many jobs finish

after the same threshold. Again, this shows the potential of using a mathematical model and a solver to generate efficient schedules, even when the obtained solution is not optimal, as in this scenario.

An important practical consideration is the amount of time required to generate a solution. As Table 3 shows, while in all scenarios where an optimal solution was found, less than 100 seconds was required (with the exception of FSM_16 and PlasticInject) to solve the respective model, in the cases where the optimal solution was not found, the model ran until the three hour time limit. Lastly, for the benchmark instances, as the instance size increases, the difficulty to solve it also increases.

In contrast, Table 3 also shows that the execution time required by STF to generate a solution was less than one second in all problem instances. Moreover, its execution time does not seem to have (strong) correlation with the problem size. This is clearly a large advantage of this approach, that quickly yields a feasible solution using a greedy procedure driven by multiple heuristics. However, its execution time depends more on the number of the competing operations and the number of heuristics and their potential combinations (a limited number of heuristics was used here) than the total number of jobs, operations and machines. Lastly, it is important to emphasize that the quality of these solutions generated by STF was, for the vast majority of instances, much lower than the quality of the solutions generated by the mathematical models.

6. Conclusion and future work

The Flexible Job Shop Scheduling Problem (F-JSSP) is a fundamental problem in modern manufacturing since solutions can provide efficient plans and schedules for assembly and production lines, reducing resource consumption and working hours. An important consideration in F-JSSP is accurately representing real world scenarios which are often very complex and specific.

This work proposes a MILP model and a MINLP model to represent F-JSSP problems that have sequence dependent setup times with start-up times and deadlines, allowing the model to better represent real-world problems. Two different and realistic objective functions are represented (makespan and tardiness). The models are used to solve 20 benchmark instances and 3 instances based on real-world scenarios. Numerical evaluation shows that the models often find optimal solutions in reasonable time for both objective functions (less than 100 seconds). In comparison to solutions generated by the STF heuristic, in all instances solutions provided by the models were better, even when not optimal. The experimental results demonstrated that the proposed model is practical and effective for solving real instances of the problem at hand.

While the models proposed in this work embody important constraints, other aspects of complex and real production lines should also be incorporated, thus allowing the models to be used in a broader context. Therefore, a suggestion for future work is the incorporation of features such as representing possible idle times after an operation (e.g., drying before folding), representing transportation time between two operations of a job in different machines, and allowing for flexible batch processing (e.g., produce 100 items in two operations with 50 items each, on separate machines).

References

- Chaudhry, I. A. e Khan, A. A. (2016). A research survey: review of flexible job shop scheduling techniques. *International Transactions in Operational Research*, 23(3):551–591. URL <https://onlinelibrary.wiley.com/doi/abs/10.1111/itor.12199>.
- Fattahi, P., Saidi Mehrabad, M., e Jolai, F. (2007). Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of intelligent manufacturing*, 18:331–342.
- Gomes, M., Barbosa-Póvoa, A., e Novais, A. (2005). Optimal scheduling for flexible job shop operation. *International Journal of Production Research*, 43(11):2323–2353. URL <https://doi.org/10.1080/00207540412331330101>.

- Kim, S. e Bobrowski, P. (1994). Impact of sequence-dependent setup time on job shop scheduling performance. *The International Journal of Production Research*, 32(7):1503–1520.
- Pinedo, M. L. (2016). *Scheduling*. Springer International Publishing. URL <http://link.springer.com/10.1007/978-3-319-26580-3>.
- Pinha, D. C., Ahluwalia, R. S., Carvalho, A. N., e Senna, P. (2015). Supply chain scheduling: A motorcycle assembly case study. *IFAC-PapersOnLine*, 48(3):1527–1532.
- Roshanaei, V., Azab, A., e ElMaraghy, H. (2013). Mathematical modelling and a meta-heuristic for flexible job shop scheduling. *International Journal of Production Research*, 51(20): 6247–6274. URL <http://www.tandfonline.com/doi/abs/10.1080/00207543.2013.827806>.
- Shen, L., Dauzère-Pérès, S., e Neufeld, J. S. (2018). Solving the flexible job shop scheduling problem with sequence-dependent setup times. *European Journal of Operational Research*, 265:503–516. URL <https://linkinghub.elsevier.com/retrieve/pii/S037722171730752X>.
- Türkyılmaz, A., Şenvar, Ö., Ünal, İ., e Bulkan, S. (2020). A research survey: heuristic approaches for solving multi objective flexible job shop problems. *Journal of Intelligent Manufacturing*, 31: 1949–1983.
- Winklehner, P. e Hauder, V. A. (2022). Flexible job-shop scheduling with release dates, deadlines and sequence dependent setup times: a real-world case. *Procedia Computer Science*, 200:1654–1663. ISSN 18770509. URL <https://linkinghub.elsevier.com/retrieve/pii/S1877050922003751>.
- Xie, J., Gao, L., Peng, K., Li, X., e Li, H. (2019). Review on flexible job shop scheduling. *IET Collaborative Intelligent Manufacturing*, 1(3):67–77.
- Xiuli, W., Junjian, P., Zirun, X., Ning, Z., e Shaomin, W. (2021). An improved multi-objective optimization algorithm for solving flexible job shop scheduling problem with variable batches. *Journal of Systems Engineering and Electronics*, 32:272–285. ISSN 1004-4132. URL <https://ieeexplore.ieee.org/document/9430105/>.
- Yeung, W.-K., Choi, T.-M., e Cheng, T. C. E. (2010). Optimal scheduling of a single-supplier single-manufacturer supply chain with common due windows. *IEEE Transactions on Automatic Control*, 55:2767–2777. ISSN 0018-9286, 1558-2523. URL <http://ieeexplore.ieee.org/document/5460987/>.