Flexible Resource Job Scheduling—A Mixed Integer Programming Approach

Vanessa Ann Beddoe Sawkmie[1,2] and Jeff Linderoth[1,2]

[1] Department of Industrial and Systems Engineering
[2] Wisconsin Institute of Discovery, University of Wisconsin-Madison
sawkmie@wisc.edu linderoth@wisc.edu

Abstract. This paper studies the minimization of makespan in job scheduling with variable machine speeds. In this problem, allocation of a resource to a machine increases its efficiency, thus reducing its processing time. We propose both a natural nonlinear disjunctive optimization model and a time-indexed mixed-integer optimization model for the problem. We show how the time-indexed model can be used to provide a lower bound on the makespan by coarsening the time discretization, and we show how to transform the coarsened solution into a feasible solution of high quality via the solution of a linear program. Computational results are presented on benchmark job-shop scheduling instances and for a commercial application with a hierarchical, two-level bill-of-materials work flow.

Keywords: machine scheduling · flexible resource allocation · mixed-integer programming

1 Introduction

The complexity and increased competition in the production sector has created the need for industries to optimize their production systems to boost manufacturing performance. Scheduling—determining the order in which tasks are processed on stations—is key to improving manufacturing efficiency. The difficulty of scheduling increases with the complexity of the manufacturing system. In most scheduling problems, machine performance is a fixed attribute. In this paper, we allow machine performance to be a decision variable. Specifically, we consider the case where there are limited resources that can be allocated to the processing stations to improve the station's performance [5,13]. The dynamic deployment of flexible resources may be considered a means of controlling the job processing times, and substantial benefits may be derived from allocating these resources and scheduling jobs concurrently. Effective utilization of resource flexibility has demonstrated significant improvements in operational performance in production systems [6,7]

In this paper, we introduce and provide mixed-integer programming (MIP) formulations for the flexible-resource job scheduling problem (FRJS). The FRJS is a generalization of the well-known (and difficult) job-shop scheduling problem

[15] to the case where a limited resource may be allocated to the processing stations to improve their efficiency. Similar flexible-resource scheduling problems have been previously studied in the literature. The parallel-machine, flexible-resource scheduling (PMFRS) problem was addressed for single-operation jobs [1]. The scheduling problem with flexible resources in a flow shop environment—where each job has a fixed precedence structure—has also been explored [3].

The standard job-shop scheduling problem without resource allocation is NP-hard even for two machines [4], and the PMFRS problem is NP-hard even for single task jobs [2]. The FRJS may have a more complicated precedence structure than PMRFS and thus is also NP-Hard. To our knowledge, FRJS has not yet been studied in the literature. In this paper, we develop two MIP models for FRJS—a natural nonlinear disjunctive model, and a time-indexed linear model.

Both models are exact formulations, but using the formulations to obtain high-quality bounds on the optimal makespan for large-scale instances is extremely challenging. In this paper, our primary focus is on comparing the effectiveness of the two models at obtaining *lower bounds* on the makespan. Using a coarse-grained time-discretization in the time-indexed model, with judicious rounding of processing times, creates a relaxation that allows us to trade-off the quality of the lower bound with solution effort. Moreover, a feasible solution, and hence upper bound on the makespan, to the scheduling problem can be extracted from the solution of the relaxation by solving a linear program.

The paper proceeds in three remaining sections. Section 2 describes the FRJS problem in detail and introduces the two MIP formulations we propose. Section 3 gives a comparison of the empirical performance of the two models in solving difficult FRJS instances, and Section 4 offers a brief conclusion of the work and avenues for additional exploration.

2 Flexible Resource Job Scheduling

2.1 Problem Description

The Flexible Resource Job Scheduling (FRJS) consists of a set I of stations and a set J of tasks to be completed on the stations. Each task $j \in J$ has a *given* station $\iota_j \in I$ on which it must be completed, and $N_i \subset J$ is the set of tasks that must be completed on station $i \in I$. There are no release times of the tasks, i.e. all tasks are available for scheduling at the beginning of the planning horizon. The set P consists of ordered task pairs. Tasks must be completed in order, and the pair of tasks $(j, k) \in P \subset J \times J$ if task j must (immediately) precede task k in the partial order. Task-scheduling is non-preemptive, i.e. once a task starts, it must be completed, and each station may only work on one task at a time.

Each task $j \in J$ has a *nominal processing time* p_j. We are given an integer amount H of a resource to distribute between the stations in I. Allocating additional resources to a station improves its efficiency and reduces the processing time of tasks completed on the station. Specifically, the *effective processing time* e_j of task j (which must be completed on station ι_j) is a function of task j's

nominal processing time p_j and the number of resources allocated to task $j's$ station. If k_{ι_j} is the number of resources allocated to station $\iota_j \in I$, then in [1], the effective processing time is defined as

$$e_j = [1 - \alpha(1 - \frac{1}{k_{\iota_j}})]p_j, \tag{1}$$

for some value of $\alpha \geq 0$. In this paper, we use $\alpha = 1$, so the processing speed on a task increases linearly with the number of resources assigned to the task's station. In Sections 2.2 and 2.3, we discuss how to implement more complicated efficiency functions.

The objective of FRJS is to allocate the resources H among the set of stations I in a way that allows to minimize the latest completion time of a task (the makespan of the schedule).

2.2 Nonlinear Disjunctive Model for FRJS

Our first MIP model for FRJS is a natural extension of the original disjunctive formulation for the job-shop scheduling problem given by Manne [10]. The model uses binary variables to enforce the logical disjunction that for each pair of tasks (j, k) that must be scheduled on a machine, that either task j finishes before k begins, or vice verse. The model requires parameters that provide an upper bound on the total processing time of each station, which we compute as

$$M_i := \beta \frac{\sum_{j \in N_i} p_j}{|I|}, \tag{2}$$

for some scaling constant $\beta > 1$. We use $\beta = 1.5$ in our instances, which is sufficient to ensure that all tasks can be completed by time M_i. We define the following decision variables:

x_j : The starting time of task $j \in J$

$$y_{jk} = \begin{cases} 1, & \text{if task } j \text{ precedes task } k \\ 0, & \text{otherwise} \end{cases} \qquad \forall i \in I; \forall (j,k) \in N_i \times N_i : j \neq k$$

v_i : Number of resources allocated to station $i \in I$

e_j : Effective processing time of task j

$\mathcal{C}_{\max}$: Makespan of schedule.

With these definitions, we can write our first (disjunctive) MIP formulation for FRJS as the following optimization problem.

$$\min \quad \mathcal{C}_{max} \tag{3a}$$

$$\text{s.t.} \quad \sum_{i \in I} v_i \leq H, \tag{3b}$$

$$e_j v_{\iota_j} \geq p_j \qquad \forall j \in J, \tag{3c}$$

$$x_k \geq x_j + e_j \qquad \forall (j,k) \in P, \tag{3d}$$

$$x_j \geq x_k + e_k - M_i y_{jk} \qquad \forall i \in I; (j,k) \in N_i \times N_i : j \neq k, \tag{3e}$$

$$x_k \geq x_j + e_j - M_i(1 - y_{jk}) \qquad \forall i \in I; (j,k) \in N_i \times N_i : j \neq k, \tag{3f}$$

$$\mathcal{C}_{max} \geq x_j + e_j \qquad \forall j \in J, \tag{3g}$$

$$x_j \geq 0 \qquad \forall j \in J, \tag{3h}$$

$$y_{jk} \in \{0,1\} \qquad \forall i \in I; (j,k) \in N_i \times N_i : j \neq k, \tag{3i}$$

$$v_i \in \mathbb{Z} \qquad \forall i \in I \tag{3j}$$

Constraint (3b) ensures the resources allocated to the stations do not exceed the amount available. Constraints (3c) ensure that the effective processing time of a task at a station is at least as large as required by equation (1) with $\alpha = 1$. This is a nonlinear constraint, but for the case $\alpha = 1$, the feasible region can be defined by a rotated second-order cone constraint and thus handled directly by most powerful commercial software packages. In the case that $\alpha \neq 1$, then, since $v_i \in \mathbb{Z}$, the relationship in (1) could be modeled using piecewise-linear variable modeling techniques [14]. Constraints (3d) enforce the precedence relationships between tasks. Constraints (3e) and (3f) are the disjunctive constraints which ensure that no two tasks are scheduled on the same station at the same time. Constraints (3g) ensure that the makespan is at least as large as the largest completion time of any task of a job. (In general, we only need to enforce this constraint for tasks at the end of any chain in the precedence structure). Constraints (3h)-(3j) enforce domain bounds on our decision variables. We denote the optimal solution value to (3) as z^*_{FRJS}.

2.3 Time-indexed Model for FRJS

The time-indexed model for FRJS is an extension of the job-shop scheduling formulation of Kondili *et al.* [8]. The key difference in the formulation is the disaggregation of the scheduling variables across the different machine configurations. Each machine $i \in I$ has a given set of potential configurations $K_i \subset \mathbb{Z}$, and operating machine $i \in I$ in configuration $k \in K_i$ requires r_{ik} of the total resource. To match the implementation of the flexible resource constraint in the nonlinear disjunctive model (3), we set $r_{ik} = k, \forall i \in I, k \in K_i$. Another key parameter for the time-indexed model is the number of time periods required to complete task j if its machine ι_j is in configuration k, which we denote as p_{jk}. We assume $p_{jk} \in \mathbb{Z}_+$. Finally, for the time-indexed model, we require a parameter T that is an upper bound on the number of periods in the scheduling horizon. The parameter T can be computed in a manner similar to (2).

Define the following decision variables:

$$z_{ik} = \begin{cases} 1, & \text{if station } i \in I \text{ is in configuration } k \in K_i \\ 0, & \text{otherwise} \end{cases}$$

$$x_{jkt} = \begin{cases} 1, & \text{if task } j \in J \text{ in station configuration } k \in K_{\iota_j} \text{ starts at the beginning of period } t \\ 0, & \text{otherwise} \end{cases}$$

Then we can write a time-index MILP model for FRJS as the following:

$$\min \quad C_{max} \tag{4a}$$

$$\text{s.t.} \quad \sum_{k \in K_i} z_{ik} = 1 \qquad \forall i \in I, \tag{4b}$$

$$\sum_{i \in I} \sum_{k \in K_i} r_{ik} z_{ik} \leq H, \tag{4c}$$

$$\sum_{t \in [T]} \sum_{k \in K_{\iota_j}} x_{jkt} = 1 \qquad \forall j \in J, \tag{4d}$$

$$\sum_{t \in [T]} x_{jkt} \leq z_{ik} \qquad \forall i \in I, j \in N_i, k \in K_{\iota_j}, \tag{4e}$$

$$\sum_{k \in K_{\iota_j}} \sum_{t \in [T]} (t + p_{jk}) x_{jkt} \leq \sum_{k \in K_{\iota_l}} \sum_{t \in [T]} t x_{lkt} \forall (j,l) \in P, \tag{4f}$$

$$\sum_{k \in K_i} \sum_{j \in N_i} \sum_{s=t-p_{jk}+1}^{t} x_{jks} \leq 1 \qquad \forall i \in I, t \in [T], \tag{4g}$$

$$\sum_{t \in [T]} \sum_{k \in K_i} (t x_{jkt} + p_{jk} z_{ik}) \leq C_{max} \qquad \forall j \in J, \tag{4h}$$

$$z_{ik} \in \{0,1\} \qquad \forall i \in I, k \in K_i, \tag{4i}$$

$$x_{jkt} \in \{0,1\} \qquad \forall j \in J, k \in K_{\iota_j}, t \in [T] \tag{4j}$$

Constraints (4b) ensure that exactly one configuration is chosen for each station. Constraint (4c) ensures that the total resources allocated to the stations do not exceed the amount available. Constraints (4d) ensure that each task starts exactly once. Constraints (4e) ensure that a task can start in a particular configuration only if that configuration is activated. Constraints (4f) enforce the precedence relationships between the required tasks. Constraints (4g) ensure that each station is running at most one task at each time point. Constraints (4h) define the makespan as the largest completion time of any task. Again, these constraints need only be written for all tasks that are at the end of a chain in the partial ordered defined by P. Constraints (4i) and (4j) define the domain bounds on our decision variables. The MIP model (4) provides an exact formulation of FRJS, so its optimal solution has value z^*_{FRJS}.

The formulation (4) can be extremely large, especially since we assume that the time periods are discretized to ensure that the parameter $p_{jk} \in \mathbb{Z}$. Preliminary computations demonstrated that the formulation was not competitive with

the nonlinear disjunctive formulation. This is not very surprising. Ku and Beck performed an extensive empirical computational comparison of MIP formulations for job shop scheduling and concluded that the disjunctive formulation was in general superior to other formulations [9].

The size of the formulation (4) can be reduced by coarsening the time discretization. Instead of assuming that each time period $t \in T$ has length 1, we can allow for the time periods to each have length $w > 1$. This naturally reduces the number of time periods T' in the transformed model $T' \leftarrow \lceil T/w \rceil$, reducing the size of the model. Increasing the time period length w also impacts the processing time parameters p_{jk}. Different choices of how to transform p_{jk} yield models with different properties. For example, rounding down the number of processing periods, rounding the number of periods to the closest integer, or rounding up the number of processing periods

$$p^{\ell}_{jk}(w) \leftarrow \left\lfloor \frac{p_{jk}}{w} \right\rfloor, \quad p^{a}_{jk}(w) \leftarrow \left\lfloor \frac{p_{jk}}{w} \right\rceil, \text{ and } \quad p^{u}_{jk}(w) \leftarrow \left\lceil \frac{p_{jk}}{w} \right\rceil \tag{5}$$

yield models whose solution values provide, respectively, a lower bound, an approximation, and an upper bound on the optimal solution value z^*_{FRJS}. In this paper, we are primarily concerned with the quality of the lower bound we are able to obtain on z^*_{FRJS}, so we confine our computational experiments to those employing p_{jk} as $p^{\ell}_{jk}(w)$ in (4).

2.4 Feasible Solutions from the Coarse-Grained Time-Indexed MIP

Solving (4) with the number of processing periods rounded down or rounded to the nearest integer, $p^{\ell}_{jk}(w)$ or $p^{a}_{jk}(w)$, may not yield a feasible solution to FRJS. Specifically, when implementing the schedule, more than one task may simultaneously be running on a station. Regardless, the solution of (4) implies an *order* for all the tasks processed on station i. With respect to a solution, define the sets

$$S_i = \{(j,k) \in N_i \times N_i \mid j \text{ immediately precedes } k \text{ on station } i \in I\}$$

as the chain of jobs processed on machine $i \in I$. Then a feasible solution to FRJS can be obtained by solving the following simple linear program:

$$\begin{aligned}
\min \quad & C_{max} && && \text{(6a)}\\
\text{s.t.} \quad & x_l \geq x_j + p_l && \forall (j,l) \in P, && \text{(6b)}\\
& x_l \geq x_j + p_j && \forall i \in I \quad \forall (j,l) \in \mathcal{S}_i, && \text{(6c)}\\
& \mathcal{C}_{max} \geq x_j + p_j && \forall j \in J. && \text{(6d)}
\end{aligned}$$

Constraints (6b) ensure that the given precedence constraints between tasks of the same job are maintained. Constraints (6c) implement the sequence order implied by the solution to the time-indexed model, and constraint (6d) ensures that the makespan is properly defined.

Our models are able to accommodate any kind of precedence structure between tasks. Some examples of different kinds of precedence structures are seen in Fig 1.

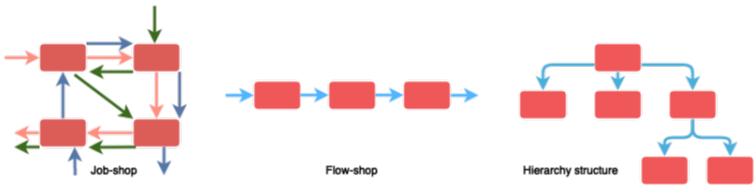

Fig. 1: Different kinds of precedence structures between tasks of jobs

2.5 Surrogate Constraints

In this work, we are primarily interested in the quality of lower bound to the optimal makespan z^*_{FRJS} that can be found by a state-of-the-art commercial MIP solver in a fixed computing effort. We have found that adding some redundant constraints based on precedence chains for tasks to the formulation (4) can typically improve the lower bound found. Let $\mathcal{C}(P)$ be a collection of chains in P, and for each chain $C \in \mathcal{C}(P)$, let A_{jC} be the set of tasks that follow task $j \in C$. For each $j \in J$, define $m_j := \min_{k \in K_{\iota_j}} p_{jk}$ as the minimum processing time for task j. For each task $j \in J$, we can be sure that it must start soon enough to ensure all of its followers in any chain are completed by the end of the planning horizon:

$$\sum_{k \in K_{\iota_j}} \sum_{t \in [T]} (t + p_{jk}) x_{jkt} \leq T - \sum_{\ell \in A_{jC}} m_\ell \quad \forall C \in \mathcal{C}(P), j \in C. \tag{7}$$

This constraint is redundant to formulation (4), but we have found that the MIP solver can often use these inequalities to generate valid inequalities that improve the lower bound. This is particularly true for job-shop problems where the partial order of preferences P can be completely partitioned into chains, so $|\mathcal{C}(P)|$ is not too large. (see Figure 1).

3 Computational Experiments

3.1 Benchmark Instances

To test the efficiency of the different formulations, computational experiments were conducted on job-shop benchmark instances `la01-la25`, `ta01-ta10`, `ft06`, `ft10`, and `ft20` from [12]. For each base job-shop instance, we generate multiple instances of our FRJS by varying H and K_i. Each base instance had two configuration sets $K_i = \{1, 2\}$ or $K_i = \{1, 2, 3\}$ for each machine and two potential choices of H for each K_i (as per table 2), resulting in four possible distinct FRJS instances from each base job-shop instance. We use a total of 135 FRJS instances in our initial testbed.

Benchmark instances	$\|I\|$	$\|J\|$
ft06	6	36
la01 - la05	5	50
la06 - la10	5	75
la11 - la15, ft20	5	100
la16 - la20, ft10	10	100
la21 - la25	10	150
ta01 - ta10	15	225

Table 1: Size of benchmark instances

$\|I\|$	K_i={1,2}	K_i={1,2,3}
5	H ={7,9}	H = {11,13}
6	H ={9,11}	H = {14,16}
10	H ={15,17}	H = {23.25}
15	H ={23,25}	H = {37,39}

Table 2: Combinations of K_i and H for instance generation

When comparing the nonlinear disjunctive formulation (3) to the time-indexed model (4), we use varying time-discretization windows w, and round down the processing times according to $p^{\ell}_{jk}(w)$ in (5), so the solution of (4) provides a lower bound on z^*_{FRJS}.

All models were implemented and run using the Python-API of Gurobi version 9.5.1. The computational experiments were run on a cluster of shared computing resources at the UW-Madison Center for High Throughput Computing. To ensure a consistent comparison between experimental results, we use Gurobi's *work unit* as the measure of computational effort. In our experience, we noted that Gurobi work units provided a consistent estimate of resource usage for an algorithm, even if run on different machines in the cluster. We allowed for an upper bound of at most 5400 work units for all computations on benchmark instances. We divide our analysis into the cases where both the nonlinear disjunctive model (3) and time-discretized model (4) were able to be solved to optimality within the work limit and cases where at least one did not solve to optimality.

3.2 Instances Solved by Both Models

A comparison of the computational performance between the set of instances solved optimally by both the disjunctive model (3), denoted by (O1), and the time-indexed model (4), denoted by (O2), within the work unit limit is shown in Fig 2.

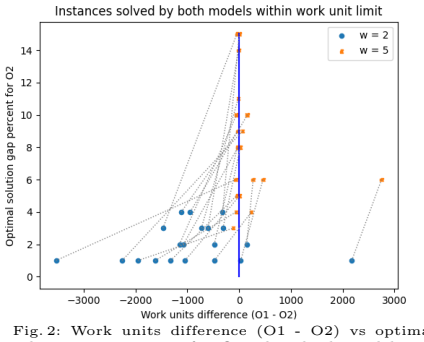

Fig. 2: Work units difference (O1 - O2) vs optimal solution gap percent for O2 when both models are solved to optimality for $w = 2$ and $w = 5$ for benchmark instances

The x-axis value plots for each instance the difference in the work units required to solve formulation (3) and model (4). Points to the left of zero represent instances where formulation (3) took a smaller number of work units to solve than model (4). For each of these 20 instances, formulation (3) was solved to optimality, so it had no optimality gap. The y-axis represents the relative optimality gap obtained between the solution of the time-indexed model 4 (for its corresponding value of w) and the optimal solution value for the instance. Each instance was solved using the time-indexed model for both $w = 2$ (in blue) and $w = 5$ (in orange). The line connecting each pair of points goes from $w = 2$ in the lower left, to $w = 5$ in the upper right, indicating that the lower bound gets smaller as we increase w, but the computing effort can often considerably improve. For the instances able to be solved by both models, clearly the nonlinear disjunctive model (3) has superior performance.

3.3 Instances Not Solved by Both Models

Fig 3 shows a computational comparison between the models on the set of instances not solved by both models. On the x-axis, we plot the optimality gap percent at the end of the work limit for the disjunctive model (3), and on the y-axis, we plot the optimality gap percent for time-indexed model (4). For each instance, regardless of the mechanism used to solve the model, we always compute the optimality gap with respect to the best known solution for the instance. Thus, this figure is comparing the lower bounds on the makespan found by the various methods.

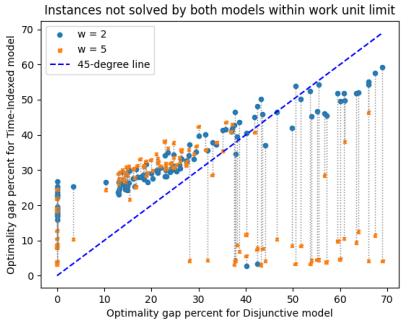

Fig. 3: Optimality gap percent of disjunctive model vs Optimality gap percent of time-indexed model for $w = 2$ and $w = 5$ for benchmark instances

The figure shows that for many of the instances, the disjunctive model has optimality gap 0 within the work limit, and the time-indexed model can have a significantly worse lower bound for these instances. We also observe comparable normalized solution gaps between the disjunctive model and time-indexed model for almost all instances for $w = 2$. This is indicated by the concentration of blue points along the 45 degree line. When we change the w parameter to $w = 5$, we see that in many instances, the optimality gap for the time-indexed model decreases considerably, indicated by the orange points in the lower right end of the graph. This indicates that the time-indexed model with an appropriate w may outperform the disjunctive model for harder instances and provide a better lower bound with the same work units limit.

3.4 Impact of Surrogate Constraint on Lower Bound

To demonstrate the positive impact that the additional redundant constraints (7) can have on the time-indexed model (4), we ran a computational experiment on the 38 base job-shop instances in Table 1, so each station had just one configuration, using the exact formulation ($w = 1$). We ran each of the 38 instances with and without inequalities (7) for a work limit of 5400 units and compared the difference in best lower bounds on z^*_{FRJS} obtained. Figure 4 displays the percentage improvement in relative lower bound for the 38 instances. For 33 of the 38 instances, the lower bound is better if inequalities (7) are added to the formulation and for some instances the improvement is quite significant.

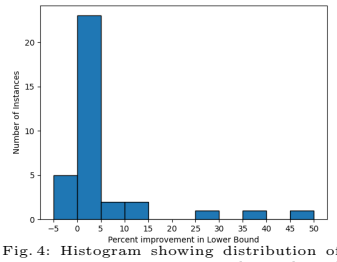

Fig. 4: Histogram showing distribution of percentage improvement in lower bound with addition of constraint 7

3.5 Performance on Commercial Data

Our commercial partner provided us with task precedences and processing times for the construction of three different types of treadmills. The production process consists of 20 machines and a hierarchical two-level bill-of-materials workflow with a final assembly task for each treadmill type. We created 135 different instances of FRJS by varying the number of each type of treadmill to produce, allowing for different machine configurations $K_i = \{1, 2\}$ or $K_i = \{1, 2, 3\}$ for each $i \in I$ and varying amount of total resource H. The resulting instances had between 112 and 554 tasks. All 135 instances were solved with the disjunctive formulation (3) and the time-indexed model (4) with time period length $w = 3$ and $w = 5$. We used an upper bound of 3600 work units for these instances.

Of the 135 instances, 87 were solved by the disjunctive formulation (3). Similar to Figure 2 for the job-shop instances, in Figure 5 we show a comparison of the computational performance between formulation (3) and (4). Similar to our benchmark instances, if the disjunctive formulation (3) is able to solve the instance, it generally does so more effectively than the time-indexed model (4).

In Figure 6 we show a computational comparison on the instances that the disjunctive formulation was not able to solve. For all these instances, the time-indexed formulation is significantly better than the disjunctive formulation.

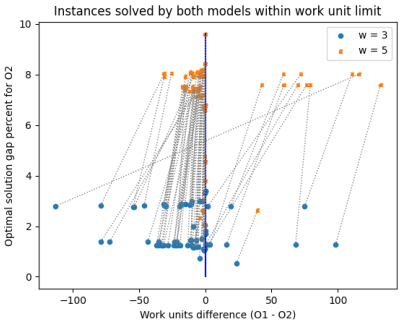

Fig. 5: Work units difference (O1-O2) vs optimal solution gap percent for O2 when both models are solved to optimality for $w = 3$ and $w = 5$ for real instances

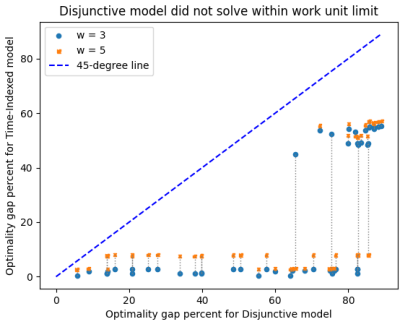

Fig. 6: Optimality gap percent of disjunctive model vs Optimality gap percent of time-indexed model for $w = 3$ and $w = 5$ for real instances

4 Conclusion

In this paper, we introduced a flexible-resource job scheduling problem and provided two mixed integer programming formulations for its solution, a natural nonlinear disjunctive formulation, and a time-indexed formulation. Coarsening the time-discretization of the time-indexed formulation gives a model than can produce a lower bound on the makespan. Computational results demonstrated that for small and medium-sized instances, the disjunctive formulation performance was superior. However, for large and difficult instances, the lower bound provided by the time-indexed formulation was often superior.

There are many avenues for future research that we plan to pursue in an expanded version of this work. First, space considerations precluded us discussing the quality of solutions obtained from the time-indexed models using the linear program (6), and we plan to investigate how to better leverage the solutions from the coarse-grained relaxation to generate high-quality feasible solutions. Second, we also plan to explore more complicated and stronger formulations of the makespan constraint, as done in [11].

References

1. Daniels, R.L., Hoopes, B.J., Mazzola, J.B.: Scheduling parallel manufacturing cells with resource flexibility. Management Science **42**(9), 1260–1276 (1996)
2. Daniels, R.L., Hoopes, B.J., Mazzola, J.B.: An analysis of heuristics for the parallel-machine flexible-resource scheduling problem. Annals of Operations Research **70**(0), 439–472 (1997)
3. Daniels, R.L., Mazzola, J.B., Shi, D.: Flow shop scheduling with partial resource flexibility. Management Science **50**(5), 658–669 (2004)
4. Garey, M.R., Johnson, D.S., Sethi, R.: The complexity of flowshop and jobshop scheduling. Mathematics of Operations Research **1**(2), 117–129 (1976)
5. Hauder, V.A., Beham, A., Raggl, S., Parragh, S.N., Affenzeller, M.: Resource-constrained multi-project scheduling with activity and time flexibility. Computers & Industrial Engineering **150**, 106857 (2020)
6. Karabati, S., Kouvelis, P., Yu, G.: The discrete resource allocation problem in flow lines. Management Science **41**(9), 1417–1430 (1995)
7. Kim, S., Nembhard, D.A.: Cross-trained staffing levels with heterogeneous learning/forgetting. IEEE Transactions on Engineering Management **57**(4), 560–574 (2010)
8. Kondili, E.M., Pantelides, C.C., Sargent, R.W.H.: A general algorithm for scheduling batch operations. In: PSE'88: Third International Symposium on Process Systems Engineering: In Affiliation with CHEMECA 88 (1988)
9. Ku, W.Y., Beck, J.C.: Mixed integer programming models for job shop scheduling: A computational analysis. Computers & Operations Research **73**, 165–173 (2016)
10. Manne, A.S.: On the job-shop scheduling problem. Operations Research **8**(2), 219–223
11. Queyranne, M.: Structure of a simple scheduling polyhedron. Mathematical Programming **58**(1), 263–285 (1993)
12. Taillard, E.: Benchmarks for basic scheduling problems. European Journal of Operational Research **64**(2), 278–285 (1993)
13. Tritschler, M., Naber, A., Kolisch, R.: A hybrid metaheuristic for resource-constrained project scheduling with flexible resource profiles. European Journal of Operational Research **262**(1), 262–273 (2017)
14. Vielma, J.P.: Mixed integer linear programming formulation techniques. SIAM Review **57**, 3–57 (2015)
15. Xiong, H., Shi, S., Ren, D., Hu, J.: A survey of job shop scheduling problem: The types and models. Computers & Operations Research **142**, 105731 (June 2022)