Reviewing
the relevant job-shop problems

Cristina Gomes Fernandes
Débora P. Ronconi
Ernesto G. Birgin
Paulo Feofiloff
Marcio Oshiro
Everton Luiz de Melo

December 1, 2012

Chapter 1

Introduction

This is an attempt to give a precise statement of the flexible job-shop scheduling problem(s) vaguely described in the paper *On-Demand Digital Print Operations: A Simulation Based Case Study* [ZJL+10, sec.2.3].

Some basic definitions and facts

A **topological ordering** of a digraph (V, A) is a permutation $(v_1, \ldots, v_n)$ of V such that $(v_i, v_j) \in A$ only if $i < j$. A **dag** is an acyclic digraph, i.e., a digraph without directed cycles. A digraph has a topological ordering if and only if it is a dag.

A digraph is a **tournament** if, for each pair (v, w) of distinct vertices, exactly one of (v, w) and (w, v) is an arc.

A directed path $(v_1, v_2, \ldots, v_n)$ in a digraph is **hamiltonian** if it contains every vertex of the digraph.

Fact: Every acyclic tournament has a (unique) hamitonian path.

A set B of arcs of a digraph is **symmetric** if, for each (v, w) in B, the arc (w, v) is also in B. A set Y of arcs is **antisymmetric** if, for each (v, w) in Y, the arc (w, v) is *not* in Y.

Chapter 2

Schedules of dags

Let (V, A) be a dag. The vertices of the dag (i.e., the elements of V) are called **operations**. Each operation v has a **processing time** p_v. We assume that p_v is a nonnegative rational number. Each arc (v, w) in A is treated as a precedence constraint: operation w cannot start before v has been completed. $p_v \geq 0$

This chapter defines the concepts of schedule, makespan, tight schedule, and critical path.

2.1 Schedules and makespan

A **schedule** for (V, A, p) is a function s from V to the set of nonnegative rational numbers such that

$$s_v + p_v \leq s_w \text{ for each } (v, w) \text{ in } A. \tag{2.1}$$

The number s_v is the **starting time** of v. (A schedule exists since (V, A) is a dag.) The **makespan** of a schedule s is the number $\mathrm{mks}(\,)$

$$\mathrm{mks}(s) := \max_{v \in V} (s_v + p_v) .$$

In order to compute a minimum-makespan schedule, we introduce next a special kind of schedule.

Tight schedule. The **length** of a (directed) path $(u_1, u_2, \ldots, u_k, u_{k+1})$ in (V, A) is the number

$$p(u_1, u_2, \ldots, u_k, u_{k+1}) := p_{u_1} + p_{u_2} + \cdots + p_{u_k} \tag{2.2}$$

(note that $p_{u_{k+1}}$ is not part of the sum).

Fact: For any path $(u_1, u_2, \ldots, u_k, u_{k+1})$ in (V, A) and any schedule s,

$$p(u_1, u_2, \ldots, u_k, u_{k+1}) \leq s_{u_{k+1}} \ . \tag{2.3}$$

We can now define a special schedule as follows: For each v in V, let s^*_v be the maximum of $p(u_1, u_2, \ldots, u_{k+1})$ over all paths $(u_1, u_2, \ldots, u_{k+1})$ such that $u_{k+1} = v$. The function s^* so defined is a schedule. We say that this is **the tight schedule** for (V, A, p). There is a simple dynamic programming algorithm that computes the tight schedule.

Not surprisingly, the makespan of the tight schedule s^* is determined by long paths: there exists a path $(u_1, u_2, \ldots, u_k, u_{k+1})$ such that

$$p(u_1, u_2, \ldots, u_k, u_{k+1}) + p_{u_{k+1}} = \mathrm{mks}(s^*) \ . \tag{2.4}$$

We say that this path is **critical**. It follows from (2.3) and (2.4) that the tight schedule has minimum makespan among all schedules for (V, A, p).

2.2 Schedules in the presence of heads and tails

For each operation v, let r_v and q_v be nonnegative numbers. We say that r_v is the **head** time (or release date) and q_v is the **tail** time of v.

An r-**schedule** for (V, A, p) is a schedule s such that

$$s_v \geq r_v$$

for each v in V. The q-**makespan** of an r-schedule s is the number

$$\mathrm{mks}_{r,q}(s) := \max_{v \in V} \left(s_v + p_v + q_v\right) \ .$$

Tight schedule. The r-**weight** of a path $(u_1, u_2, \ldots, u_k, u_{k+1})$ in (V, A, p) is defined by the recursion

$$\mathrm{wt}_r(u_1, u_2, \ldots, u_k, u_{k+1}) = \max\left(\mathrm{wt}_r(u_1, u_2, \ldots, u_k) + p_{u_k},\ r_{u_{k+1}}\right) \ .$$

(Thus, for example, $\mathrm{wt}_r(u_1) = r_{u_1}$ and $\mathrm{wt}_r(u_1, u_2) = \max(r_{u_1} + p_{u_1},\ r_{u_2})$.) It follows from this definition that

$$\mathrm{wt}_r(u_1, u_2, \ldots, u_k, u_{k+1}) = \max_{i=1}^{k+1} \left(r_{u_i} + p_{u_i} + \cdots + p_{u_k}\right) \ . \tag{2.5}$$

Fact: For any path $(u_1, u_2, \ldots, u_k, u_{k+1})$ in (V, A) and any r-schedule s,

$$\mathrm{wt}_r(u_1, u_2, \ldots, u_k, u_{k+1}) \leq s_{u_{k+1}} \,. \tag{2.6}$$

For each v in V, let s^*_v be the maximum of $\mathrm{wt}_r(u_1, u_2, \ldots, u_k, u_{k+1})$ over all paths $(u_1, u_2, \ldots, u_k, u_{k+1})$ such that $u_{k+1} = v$. The function s^* so defined is a r-schedule. We say that this is **the tight r-schedule** for (V, A, p). There is a simple dynamic programming algorithm that computes the tight r-schedule.

The q-makespan of the tight r-schedule s^* is equal to the r-weight of a heavy path: there is a path $(u_1, u_2, \ldots, u_{k+1})$ such that

$$\mathrm{wt}_r(u_1, u_2, \ldots, u_{k+1}) + p_{u_{k+1}} + q_{u_{k+1}} = \mathrm{mks}_{r,q}(s^*) \,.$$

In view of (2.5), this can be rephrased as follows: there is a path $(u_i, u_{i+1}, \ldots, u_{k+1})$ such that

$$r_{u_i} + p_{u_i} + \cdots + p_{u_k} + p_{u_{k+1}} + q_{u_{k+1}} = \mathrm{mks}_{r,q}(s^*) \,.$$

Such a path is said to be **critical**. In view of (2.6), the tight r-schedule has minimum q-makespan among all r-schedules for (V, A, p).

2.3 Schedules and lead time

Makespan is not the only parameter to measure the "quality" of a schedule. We describe next another such parameter.

Suppose we have a dag (V, A) with processing times p and head times r, but without tail times (i.e., $q = 0$). The **maximum lead time** of an r-schedule s for (V, A, p) is the number

$$\mathrm{mlt}(s) := \max_{v \in V} (s_v - r_v + p_v) \,.$$

Similarly, the **sum of lead times** (also known as the **flow time**) is the number

$$\mathrm{slt}(s) := \textstyle\sum_{v \in V} (s_v - r_v + p_v) \,.$$

How does one find a schedule that minimizes mlt? How does one find a schedule that minimizes slt? For the second problem, it seems reasonable to put the operations in increasing order of $r_v + p_v$. This solves the problem if $r_{v_{i-1}} + p_{v_{i-1}} \leq r_{v_i}$ for all i (in which case $\mathrm{slt}(s) = p_{v_1} + \cdots + p_{v_n}$, where $n = |V|$). It also solves the problem if $r_i = 0$ for all i (in which case $\mathrm{slt}(s) = np_{v_1} + (n-1)p_{v_2} + \cdots + 1p_{v_n}$). But it does not work in general...

Chapter 3

Ordinary job-shop scheduling

We begin with the ordinary job-shop scheduling (JSS) problem, without the "flexible" attribute and without heads and tails.

Our formulation of the JSS problem is somewhat unusual, because it has no explicit concept of a job. But it is simpler and cleaner than the standard formulation.

3.1 A statement of the JSS problem

V We are given a dag (V, A). The vertices of the dag (i.e., the elements of V) are called **operations**. (The weakly connected components of the dag may be called
A **jobs**.) Each arc (v, w) of the dag is a **hard precedence constraint**.

p_v For each operation v, there is a rational number p_v that represents the **processing time** of v. We assume that

$$p_v > 0 \, .$$

(Forbidding null values simplifies the presentation, since in this case a digraph is a dag if and only if it has a schedule.)

m We are also given a set of **machines** numbered $1, \ldots, m$ and a function f that
f assigns a machine $f(v)$ to each operation v.

B_k For each machine k, let B_k be the set of all ordered pairs of distinct elements of
B $\{v \in V : f(v) = k\}$. Of course B_k is symmetric. Let

$$B := B_1 \cup \cdots \cup B_m \, .$$

Each (v, w) in B is a **soft arc** or **soft precedence constraint** or **intra-machine precedence constraint**, and is designed to prevent two different operations to be pro-

cessed on the same machine at the same time. The digraph (V, B) is known as **disjunctive graph**.

A **selection** is a subset Y of B such that, for each (v, w) in B, exactly one of (v, w) and (w, v) is in Y. In other words, Y is a selection if Y

$$Y \text{ and } B - Y \text{ are both antisymmetric.}$$

A selection Y is **good** if $(V, A \cup Y)$ is a dag. Any good selection can be represented by a topological ordering $(v_1, \ldots, v_n)$ of (V, A): an ordered pair (v_i, v_j) belongs to the selection if and only if $i < j$ and $f(v_i) = f(v_j)$.

The **makespan** of a good selection Y is the makespan of the tight schedule (see section 2.1) for $(V, A \cup Y, p)$. The makespan of Y is denoted by $\mathrm{mks}(Y)$. $\mathrm{mks}(Y)$

> JSS PROBLEM (V, A, p, f): Find a good selection Y that has minimum makespan.

The problem is NP-hard.

3.2 A MILP formulation of the problem

Here is a MILP formulation of the JSS problem. A binary array y indexed by B will represent a selection: we shall have $y_{v,w} = 1$ if and only if (v, w) belongs to the selection. We also need an upper bound L on the makespan of an optimal solution. (This can be the makespan of some good selection or, alternatively, the sum $\sum_{v \in V} p_v$.) y L

The following MILP is equivalent to the JSS problem: find a rational array s and a binary array y that minimize an auxiliary rational variable z under the following linear constraints: s z

$$z \leq L, \tag{3.1}$$
$$s_v + p_v \leq z \quad \text{f.e. } v \text{ in } V, \tag{3.2}$$
$$s_v \geq 0 \quad \text{f.e. } v \text{ in } V, \tag{3.3}$$
$$0 \leq y_{v,w} \leq 1 \quad \text{f.e. } (v, w) \text{ in } B, \tag{3.4}$$
$$y_{v,w} + y_{w,v} = 1 \quad \text{f.e. } (v, w) \text{ in } B, \tag{3.5}$$
$$s_v + p_v - (1 - y_{v,w})L \leq s_w \quad \text{f.e. } (v, w) \text{ in } B, \tag{3.6}$$
$$s_v + p_v \leq s_w \quad \text{f.e. } (v, w) \text{ in } A, \tag{3.7}$$

where "f.e." means "for each".

The correspondence between our MILP and the JSS problem is easy to check. Suppose Y is an optimal solution of the JSS problem. Let $z := \mathrm{mks}(Y)$. Let s be the tight schedule for $(V, A \cup Y, p)$. For each (v, w) in B, let $y_{v,w} = 1$ if $(v, w) \in Y$ and $y_{v,w} = 0$ otherwise. Then (s, y, z) satisfies constraints (3.1) through (3.7).

Now consider the converse. Let (s, y, z) be an optimal solution of the MILP. Let Y be the set of all pairs (v, w) in B for which $y_{v,w} = 1$. Constraints (3.4) and (3.5) make sure that Y is a selection. Since $p_v > 0$, constraints (3.6) and (3.7) make sure that Y is a good selection (and s is a schedule). The minimality of z implies the minimality of $\mathrm{mks}(Y)$. Hence, Y is a solution of the JSS problem.

The MILP has $4|B| + |A| + 2|V| + 1$ constraints and $|B| + |V| + 1$ variables, of which only $|B|$ are integer.

3.3 Lower bounds and valid inequalities

A **lower bound** for the JSS problem (as well as for its MILP) is any number $\underline{z}$ such that $\mathrm{mks}(Y) \geq \underline{z}$ for any good selection Y. Obtaining good lower bounds is an important step in many heuristics for the JSS problem

Take the linear relaxation of the MILP (3.1) through (3.7) (i.e., drop the requirement that y be integral) and let $(\dot{s}, \dot{y}, \dot{z})$ be a solution of the resulting linear programming problem. Of course $\dot{z}$ is a lower bound for the MILP. In order to get the tighter lower bound, we can add new inequalities to the linear programming problem that are redundant for the MILP but not for its relaxation. This pool of **valid inequalities** can be used in a branch-and-cut process as follows:

1. If $\dot{y}$ is integral, then stop.
2. If $\dot{y}$ is not integral, then
   - 2a. find (by exhaustive search) a violated valid inequality;
   - 2b. add the violated inequality to the linear programming program;
   - 2c. let $(\dot{s}, \dot{y}, \dot{z})$ be a solution of the new program;
   - 2d. return to step 1.

The lower bound $\dot{z}$ will get tighter with each iteration.

In order to formulate the valid inequalities we must introduce two new pieces of data: the heads and the tails. These are defined as follows.

r_v

- For each v, let r_v be the length (see section 2.1) of a longest path in (V, A) terminating at v. According to (2.3), for any good selection Y and any schedule s for $(V, A \cup Y, p)$, we shall have $s_v \geq r_v$ for each operation v.

- For each v, let q_v be the greatest sum of the form $p_{u_2}+p_{u_2}+\cdots+p_{u_{k+1}}$ (note that p_{u_1} is not part of the sum) for any path $(u_1, u_2, \ldots, u_k, u_{k+1})$ in (V, A) starting at v. For any good selection Y and any schedule s for $(V, A \cup Y, p)$, we shall have $\mathrm{mks}(s) - (s_v + p_v) \geq q_v$ for each operation v. q_v

Applegate and Cook [AC91, BDP96] experimented with the following valid inequalities:

Basic cuts. For any machine k and subset W of $\{v \in V : f(v) = k\}$,

$$\sum_{w\in W} p_w s_w \geq (\sum_{w\in W} p_w)(\min_{v\in W} r_v) + \tfrac{1}{2}\sum_{(v,w)\in C} p_v p_w \;, \tag{3.8}$$

where C is the set of all ordered pairs of distinct elements of W (of course, $C \subseteq B_k$). To show that this is a valid inequality, let Y be a good selection and s a schedule for $(V, A \cup Y, p)$. Then $(W, Y \cap C)$ is an acyclic tournament. It follows that, for each w in W, $s_w \geq \min_{v\in W} r_v + \sum_{(v,w)\in Y\cap C} p_v$. (See section 2.2.) Multiply each such inequality by p_w and add the results together to obtain (3.8).

The inequality (3.8) reduces to $p_v s_v + p_w s_w \geq (p_v + p_w)\min(r_v, r_w) + p_v p_w$ when $W = \{v, w\}$ and $C = \{(v, w), (w, v)\}$.

Reverse basic cuts. This is a reverse version of (3.8): for any machine k and subset W of $\{v \in V : f(v) = k\}$,

$$\sum_{v\in W} p_v(z - s_v - p_v) \geq (\sum_{v\in W} p_v)(\min_{w\in W} q_w) + \tfrac{1}{2}\sum_{(v,w)\in C} p_v p_w \;, \tag{3.9}$$

where C is the set of all ordered pairs of distinct elements of W. When $W = \{v, w\}$ and $C = \{(v, w), (w, v)\}$, the inequality reads as follows: $p_v(z - s_v - p_v) + p_w(z - s_w - p_w) \geq (p_v + p_w)\min(q_v, q_w) + p_v p_w$.

Two-job inequalities. For each (v, w) in B,

$$(p_v + r_v - r_w)s_v + (p_w + r_w - r_v)s_w \geq p_v p_w + r_v p_w + r_w p_v \;. \tag{3.10}$$

To show that this is a valid inequality, let α and β be nonnegative parameters. Let Y de any good selection Y and s any schedule for $(V, A \cup Y, p)$. If $(v, w) \in Y$ then $\alpha s_v + \beta s_w \geq \alpha r_v + \beta(r_v + p_v)$. If $(w, v) \in Y$ then $\alpha s_w + \beta s_v \geq \alpha r_w + \beta(r_w + p_w)$. Both inequalities will be simultaneously satisfied if $\alpha = p_v + r_v - r_w$ and $\beta = p_w + r_w - r_v$. Inequality (3.10) follows.

If $r_v + p_v > r_w$ and $r_w + p_w > r_v$, the inequality (3.10) is stronger than the inequalities (3.8) and (3.9) for the case $W = \{v, w\}$ and $C = \{(v, w), (w, v)\}$.

Triangle cuts. For any three distinct operations u, v, w, if (u, v), (v, w) and (w, u) are in B then

$$y_{u,v} + y_{v,w} + y_{w,u} \leq 2 \;.$$

These inequalities are valid because, for any good selection Y, the digraph $(V, A \cup Y)$ is a dag.

Half cuts. For each machine k, subset W of $\{w \in V : f(w) = k\}$, and operation w in W,

$$s_w \geq \min_{v \in W} r_v + \sum_{(v,w) \in C} y_{v,w} p_v \,, \tag{3.11}$$

where C is the set of all ordered pairs of distinct elements of W. These inequalities are called "half cuts" because they are "uselful for pushing the value of the y's away from $1/2$" [AC91].

Chapter 4

One-machine scheduling with heads and tails

The set of instances of the JSS problem for which $m = 1$ is known as **one-machine subproblem**. This subproblem is trivial since all good selections will have the same makespan, equal to $\sum_{v \in V} p_v$. The one-machine subproblem becomes non-trivial, however, if we add head times and tail times (see section 2.2) to its statement.

4.1 One machine, with heads and tails

Suppose each operation v has a head r_v and a tail q_v (both are nonnegative rational numbers). Assume that $A = \emptyset$ and $m = 1$, whence B is the set of all ordered pairs of distinct vertices. A good selection, in this case, is a subset Y of B such that Y and $B - Y$ are both antisymmetric and the digraph (V, Y) is a dag. The makespan of a good selection Y is the q-makespan of the tight r-schedule for (V, Y, p). The one-machine subproblem asks for a good selection Y that minimizes the makespan of Y.

This variant of the one-machine subproblem is NP-hard, but Carlier [Car82] has a very effective branch-and-bound heuristic for it. The branch-and-bound uses Schrage's [Sch71] algorithm (see figure 4.1), to produce a good selection Y for which $\mathrm{mks}(Y)$ is relatively small. Specifically,

$$\mathrm{mks}(Y) < \mathrm{mks}(Y_*) + \max_{v \in V} p_v \,, \tag{4.1}$$

where Y_* is an optimal solution of the problem (and therefore $\mathrm{mks}(Y) \geq \mathrm{mks}(Y_*)$). The proof of (4.1) depends on the following inequality: For any nonempty $X \subseteq V$

and any good selection Y,

$$\varphi(X) \leq \mathrm{mks}(Y)\,, \tag{4.2}$$

where

$$\varphi(X) := \min_{i\in X} r_i + \sum_{i\in X} p_i + \min_{i\in X} q_i\,.$$

In particular, $\max_{\emptyset\subset X\subseteq V} \varphi(X) \leq \min_Y \mathrm{mks}(Y)$ (but the inequality is usually strict).

An extension of inequality (4.2) can be stated as follows: For any $X \subseteq V$, any $w \in V - X$, and any good selection Y, if $\{w\} \times X \subseteq Y$ or $X \times \{w\} \subseteq Y$ then $\varphi(X) + p_w \leq \mathrm{mks}(Y)$. The proof of (4.1) boils down to the following

Lemma (Carlier [Car82]): Let Y be the selection produced by Schrage's algorithm. Then either (1) there exists $X \subseteq V$ such that $\varphi(X) = \mathrm{mks}(Y)$ (in which case Y is optimal) or (2) there exist $X \subseteq V$ and $y \in V - X$ such that $\mathrm{mks}(Y) < \varphi(X) + p_y$ and, in any optimal selection Y_*, either $\{y\} \times X \subseteq Y_*$ or $X \times \{y\} \subseteq Y_*$.

```
SCHRAGE (V, p, r, q)
1   Y ← ∅
2   t ← 0
3   U ← V
4   while U ≠ ∅ do
5       t' ← min_{v∈U} r_v
6       if t < t' then t ← t'
7       U' ← {v ∈ U : r_v ≤ t}
8       choose v in U' that maximizes q_v
9       t ← t + p_v
10      U ← U − {v}
11      Y ← Y ∪ ({v} × U)
12  return Y
```

Figure 4.1: Schrage's algorithm. It receives (V, p, r, q) and returns a good selection Y such that $\mathrm{mks}(Y) < \mathrm{mks}(Y_*) + \max_{v\in V} p_v$, where Y_* is an optimal solution of the problem.

Computing the lower bound φ. The lower bound $\max_X \varphi(X)$ for $\mathrm{mks}(Y_*)$ is easy to compute: algorithm LOWERBOUNDPHI does the job (see figure 4.2). This lower bound that can be very useful in a branch-and-bound scheduling algorithm.

```
LOWERBOUNDPHI (V, p, r, q)
1  l ← −∞
2  for each ordered pair (v, w) of elements of V do    ▷ also v = w
3      X ← {x ∈ V : r_x ≥ r_v and q_x ≥ q_w}
4      if v ∈ X and w ∈ X
5          then  s ← Σ_{x∈X} p[x]
6                l ← max (l, r_v + s + q_w)
7  return l
```

Figure 4.2: Returns the number $l := \max_X \varphi(X)$, where the maximum is taken over all nonempty subsets X of V and $\varphi(X)$ is $\min_{x\in X} r_x + \sum_{x\in X} p_x + \min_{x\in X} q_x$. This is a lower bound on $\mathrm{mks}(Y_*)$, where Y_* is an optimal solution of the problem.

Chapter 5

Flexible job-shop scheduling

The **flexible** job-shop scheduling (F-JSS) problem is a generalization of the JSS problem in which each operation can be processed on any machine of a given set of machines.

5.1 Statement of F-JSS problem

V
A

We are given a dag (V, A). Its vertices are called **operations** and its arcs are called **hard precedence constraint**.

m
F

We are also given a set of **machines** numbered $1, \ldots, m$ and a function F that associates a nonempty subset $F(v)$ of $\{1, \ldots, m\}$ with each operation v.

$p_{v,k}$

For each operation v and each machine k in $F(v)$, there is a rational number $p_{v,k}$ that represents the **processing time** of operation v on machine k. We assume that

$$p_{v,k} > 0\,,$$

since forbidding null processing times simplifies the presentation.

V_k
B_k
B

For each machine k, let $V_k := \{v \in V : F(v) \ni k\}$ (this is the set of all operations that can be processed on machine k) and let B_k be the set of all ordered pairs of distinct elements of V_k. Let B denote the union of all B_k. Of course B is symmetric. Each (v, w) in B is a **soft arc** or **soft precedence constraint**.

A **machine assignment** is a function f that assigns a machine to each operation so that

$$f(v) \in F(v)$$

B^f

for each v. Given a machine assignment f, let B^f be the set of all ordered pairs of

distinct vertices to be processed on the same machine:

$$B^f := \{(v, w) \in B : f(v) = f(w)\} .$$

For a given machine assignment f, a **selection** is any subset Y of B^f such that

$$Y \text{ and } B^f - Y \text{ are both antisymmetric,}$$

i.e., for each (v, w) in B^f, exactly one of (v, w) and (w, v) is in Y.

A selection Y is **good** if $(V, A \cup Y)$ is a dag. Any good selection can be represented by a topological ordering $(v_1, \ldots, v_n)$ of (V, A): an ordered pair (v_i, v_j) belongs to the selection if and only if $i < j$ and $f(v_i) = f(v_j)$.

The **makespan** of a good selection Y (for a given machine assignment f) is the makespan of the tight schedule (see section 2.1) for $(V, A \cup Y, p')$, where $p'_v = p_{v,f(v)}$ for each v. We denote the makespan of Y by $\mathrm{mks}(Y)$. $\mathrm{mks}(Y)$

F-JSS PROBLEM (V, A, p, m, F): Find a machine assignment f and a good selection Y such that $\mathrm{mks}(Y)$ is minimum.

5.2 Our MILP formulation of the F-JSS problem

In order to formulate the F-JSS problem as a MILP, we use a binary array x to represent the machine assignments and a binary array y to represent selections. The first array will have a component $x_{v,k}$ for each v in V and each k in $F(v)$. The second will have a component $y_{v,w}$ for each (v, w) in B. x y

Our MILP needs an upper bound L on the makespan of an optimal solution of the F-JSS problem. This can be the makespan of an arbitrary good selection or, alternatively, a global bound like $\sum_{v \in V} \max_{k \in F(v)} p_{v,k}$. L

Finally, we shall use an auxiliary array p' to store the processing times of the operations corresponding to the machine assignment x. p'

Our MILP can now be formulated as follows: find rational arrays s and p' and *binary* arrays x and y that minimize an auxiliary rational variable z under the s z

following linear constraints:

$$s_v + p'_v \le z \qquad \text{f.e. } v \text{ in } V, \tag{5.1}$$
$$s_v \ge 0 \qquad \text{f.e. } v \text{ in } V, \tag{5.2}$$
$$0 \le x_{v,k} \le 1 \qquad \text{f.e. } v \text{ in } V \text{ and } k \text{ in } F(v), \tag{5.3}$$
$$\textstyle\sum_{k \in F(v)} x_{v,k} = 1 \qquad \text{f.e. } v \text{ in } V, \tag{5.4}$$
$$p'_v = \textstyle\sum_{k \in F(v)} p_{v,k} x_{v,k} \qquad \text{f.e. } v \text{ in } V, \tag{5.5}$$
$$0 \le y_{v,w} \le 1 \qquad \text{f.e. } (v,w) \text{ in } B, \tag{5.6}$$
$$y_{v,w} + y_{w,v} \ge x_{v,k} + x_{w,k} - 1 \qquad \text{f.e. } k \text{ in } \{1,\dots,m\} \text{ and } (v,w) \text{ in } B_k, \tag{5.7}$$
$$s_v + p'_v - (1 - y_{v,w})L \le s_w \qquad \text{f.e. } (v,w) \text{ in } B, \tag{5.8}$$
$$s_v + p'_v \le s_w \qquad \text{f.e. } (v,w) \text{ in } A, \tag{5.9}$$

where the expression "f.e." means "for each".

We show next that our MILP is equivalent to the F-JSS problem. Suppose (f, Y) is an optimal solution of the F-JSS problem. Let $p'_v := p_{v,f(v)}$ for each v. Let $z := \mathrm{mks}(Y)$. Let s be the tight schedule for $(V, A \cup Y, p')$. By definition, $\mathrm{mks}(Y) = \mathrm{mks}(s)$. For each operation v and each k in $F(v)$, let $x_{v,k} = 1$ if and only if $f(v) = k$. Finally, let $y_{v,w} = 1$ for each (v,w) in Y and $y_{v,w} = 0$ for each (v,w) in $B - Y$. Then the tuple (s, x, p', y, z) satisfies constraints (5.1) through (5.9).

Now consider the opposite transformation. Let (s, x, p', y, z) be an optimal solution of our MILP. For each v, let $f(v)$ be the unique k in $F(v)$ for which $x_{v,k} = 1$; such k exists by virtue of constraints (5.3) and (5.4). According to (5.5), $p'_v = p_{v,f(v)}$. Let Y be the set of all pairs (v,w) in B such that $y_{v,w} = 1$. Constraint (5.7) makes sure that Y is a selection. Since $p_{v,k} > 0$ for all (v,k), constraints (5.8) and (5.9) make sure that the selection Y is good (and s is a schedule for $(V, A \cup Y, p')$). Hence, (f, Y) is a solution of the F-JSS problem. Moreover, $z = \mathrm{mks}(Y)$ due to the minimality of z.

Our MILP has $4|V| + \sum_v |F(v)| + 2|B| + \sum_{k=1}^{m} |B_k| + |A|$ constraints and $|B| + \sum_v |F(v)| + 2|V| + 1$ variables, of which only $|B| + \sum_v |F(v)|$ are integer. (Hence, the number of constraints is bounded by $4n + n\gamma + 2n^2 + mn^2 + |A|$ and the number of variables is bounded by $n^2 + n\gamma + 2n + 1$, where n is the number of operations and $\gamma := \max_v |F(v)|$.)

A good property of our MILP. Consider the linear relaxation of our MILP, i.e., drop the requirement that x and y be integer. Let $(\dot{s}, \dot{x}, \dot{p}', \dot{y}, \dot{z})$ be an optimal solution of the resulting LP. By virtue of (5.1), (5.2) and (5.9), $\dot{s}$ is a schedule for $(V, A, \dot{p}')$ and $\mathrm{mks}(\dot{s}) \le \dot{z}$.

Define p'' by setting $p''_v := \min_{k \in F(v)} p_{v,k}$ for each v. Let s'' be the tight schedule for (V, A, p''). Since $p'' \le \dot{p}'$ (because $\dot{p}'_v$ is a convex combination of the $p_{v,k}$), we have

$\mathrm{mks}(s'') \leq \mathrm{mks}(\dot{s})$. Therefore, $\mathrm{mks}(s'')$ is a lower bound on $\dot{z}$. The existence of such reasonable lower bound on $\dot{z}$ is a good property of our MILP, a property that the ÖÖY MILP to be described in section 5.5 does not have.

5.3 Testing our MILP

We applied our MILP (section 5.2) to a number of instances of the F-JSS problem found in the literature (see figure 5.1). Instances SFJS01 to SFJS10 and MFJS01 to MFJS10 were kindly provided by Fattahi *et al.* [FMJ07]. Instances MK01 to MK15 were taken from Mastrolilli [Mas], who got them from Brandimarte [Bra93].

instance	n	$\lvert A\rvert$	m
SFJS01	4	2	2
SFJS02	4	2	2
SFJS03	6	3	2
SFJS04	6	3	2
SFJS05	6	3	2
SFJS06	9	6	3
SFJS07	9	6	5
SFJS08	9	6	4
SFJS09	9	6	3
SFJS10	12	8	5
MFJS01	15	10	6
MFJS02	15	10	7
MFJS03	18	12	7
MFJS04	21	14	7
MFJS05	21	14	7
MFJS06	24	16	7
MFJS07	32	24	7
MFJS08	36	27	8
MFJS09	44	33	8
MFJS10	48	36	8

instance	n	$\lvert A\rvert$	m
MK01	55	45	6
MK02	58	48	6
MK03	150	135	8
MK04	90	75	8
MK05	106	91	4
MK06	150	140	15
MK07	100	80	5
MK08	225	205	10
MK09	240	220	10
MK10	240	220	15
MK11	179	149	5
MK12	193	163	10
MK13	231	201	10
MK14	277	247	15
MK15	284	254	15

Figure 5.1: Test instances of the F-JSS problem. The instances on the left are from Fattahi *et al.* while those on the right came from Brandimarte. Columns: n is the number of operations, $|A|$ is the number of hard precedence constraints, and m is the number of machines.

The result of applying our MILP to the test instances is recorded in figure 5.2. Figures 5.3 and 5.4 show the evolution of the lower and upper bounds on the minimum makespan as a branch-and-bound algorithm tries to solve our MILP.

instance	mks	time
SFJS01	66	0.0
SFJS02	107	0.0
SFJS03	221	0.0
SFJS04	355	0.0
SFJS05	119	0.0
SFJS06	320	0.0
SFJS07	397	0.0
SFJS08	253	0.0
SFJS09	210	0.0
SFJS10	516	0.0
MFJS01	468	0.3
MFJS02	446	0.5
MFJS03	466	1.4
MFJS04	554	17.
MFJS05	514	3.
MFJS06	634	35.
MFJS07	879	1370.
MFJS08	764 - 884	3600.
MFJS09	818 - 1088	3600.
MFJS10	944 - 1258	3600.

instance	mks
MK01	39 - 40
MK02	25 - 29
MK03	92 - 231
MK04	40 - 63
MK05	63 - 190
MK06	37 - 78
MK07	67 - 206
MK08	181 - 542
MK09	146 - 455
MK10	-
MK11	159 - 705
MK12	180 - 540
MK13	-
MK14	232 - 694
MK15	-

Figure 5.2: Solving test instances (see figure 5.1) of the F-JSS problem. Column mks gives the minimum makespan computed by the CPLEX software [CPL12] running our MILP. (For the MK instances, only lower and upper bounds on the minimum — if that much — were found.) For instances SFJS01 to MFJS10, the time is given in seconds. Instances MK01 to MK15, took 1 hour each. We used an Intel Xeon server with one 2.83GHz core and 32GB RAM.

5.4 Valid inequalities for our MILP

We record next a few valid inequalities for our MILP, i.e., inequalities that are redundant for the MILP but not for its linear relaxation:

$$y_{v,w} + y_{w,v} \leq 1 \qquad \text{f.e. } (v,w) \text{ in } B, \tag{5.10}$$

$$y_{u,v} + y_{v,w} + y_{w,u} \leq 2 \qquad \text{f.e. } (u,v), (v,w), \text{ and } (w,u) \text{ in } B, \tag{5.11}$$

$$y_{v,w} \leq 1 - x_{v,k} + x_{w,k} \qquad \text{f.e. } k \text{ in } \{1 \ldots, m\} \text{ and } (v,w) \text{ in } B_k, \tag{5.12}$$

$$y_{v,w} \leq 1 + x_{v,k} - x_{w,k} \qquad \text{f.e. } k \text{ in } \{1 \ldots, m\} \text{ and } (v,w) \text{ in } B_k. \tag{5.13}$$

Constraints (5.12) and (5.13) make sure that $y_{v,w}$ is 0 whenever one of $x_{v,k}$ and $x_{w,k}$ is 1 and the other is 0.

In principle, these inequalities could speed up the solution our MILP. But some experimentation with test data suggests that they do not help.

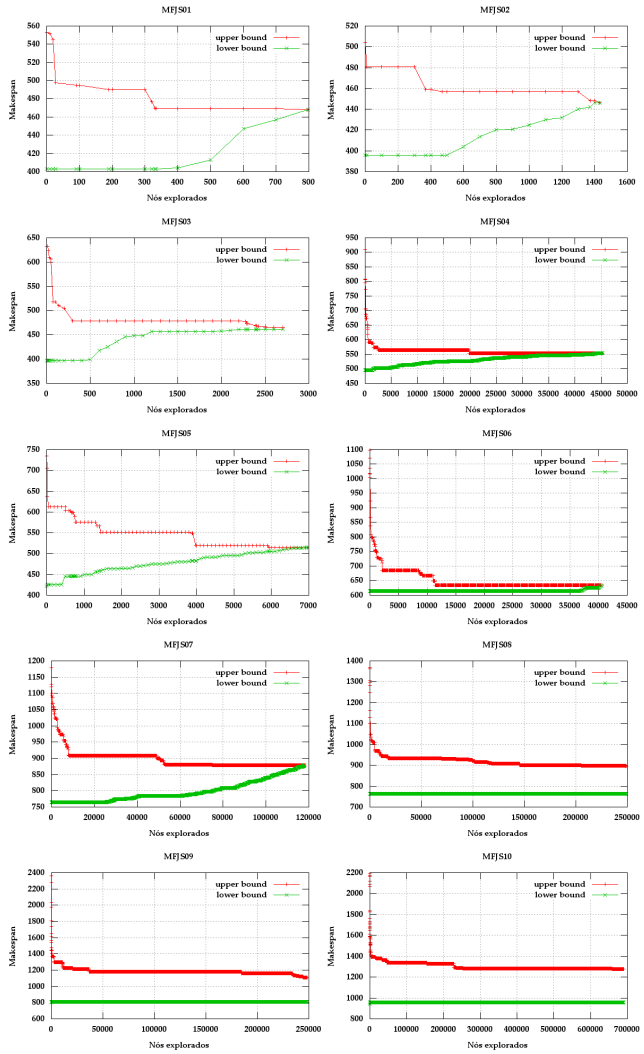

Figure 5.3: Trying to solve the MFJS test instances using our MILP (see figure 5.2). The graphs show the lower and upper bounds on the minimum makespan as the CPLEX branch-and-bound algorithm progresses towards the solution. (The graphs can be better seen at www.ime.usp.br/~oshiro/flexjobshop/iranianos/mip/fattahietal.png.)

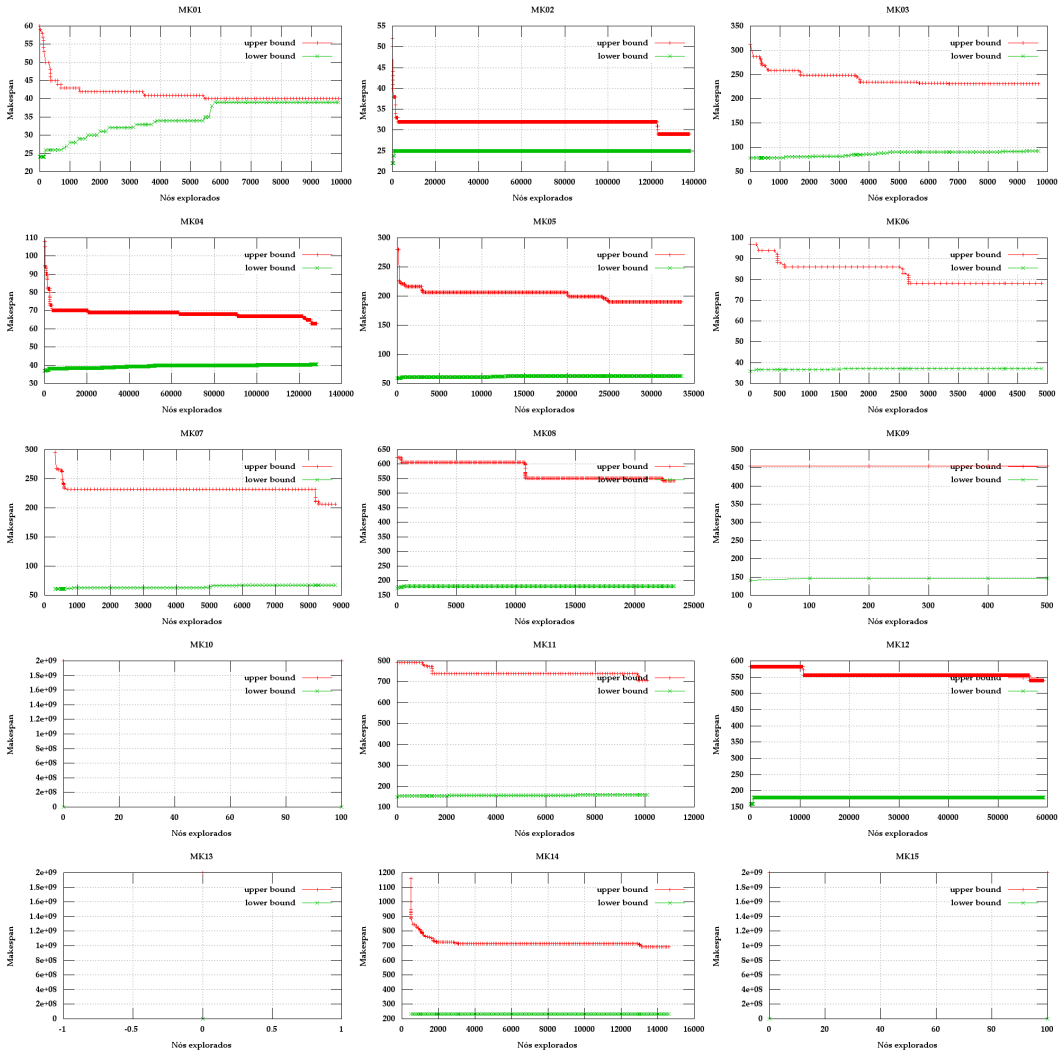

Figure 5.4: Trying to solve the MK test instances using our MILP (see figure 5.2). The graphs show the lower and upper bounds on the minimum makespan as the CPLEX branch-and-bound algorithm progresses towards the solution. (The graphs can be better seen at www.ime.usp.br/~oshiro/flexjobshop/brandimarte/brandimarte.png.)

5.5 The ÖÖY MILP for the F-JSS problem

Özgüven–Özbakir–Yavuz [ÖÖrY10], building on the work of Fattahi *et al.* [FMJ07], give a different (and less natural) MILP for the F-JSS problem. We state next a slightly reorganized version of the ÖÖY MILP. We shall use the following variables:

- A binary array x to represent machine assignments, with one component $x_{v,k}$ for each v in V and each k in $F(v)$. x
- A rational array t to represent the starting times,with one component $t_{v,k}$ for each v in V and each k in $F(v)$. t
- A rational array c to represent the completion times, with one component $c_{v,k}$ for each v in V and each k in $F(v)$. c
- A binary array y to represent a selection, with one component $y_{v,w,k}$ for ewach k in $\{1,\ldots,m\}$ and (v,w) in B_k. y

We also need an upper bound L on the makespan of an optimal solution. (This can be the makespan of some good selection or the sum $\sum_{v\in V} \max_{k\in F(v)} p_{v,k}$, for example.) L

The ÖÖY MILP is as follows: find rational arrays t and c and binary arrays x and y that minimize an auxiliary rational variable z under the following linear constraints:

$$c_{v,k} \le z \qquad \text{f.e. } v \text{ in } V \text{ and } k \text{ in } F(v), \tag{5.14}$$

$$t_{v,k} \ge 0 \qquad \text{f.e. } v \text{ in } V \text{ and } k \text{ in } F(v), \tag{5.15}$$

$$c_{v,k} \ge 0 \qquad \text{f.e. } v \text{ in } V \text{ and } k \text{ in } F(v), \tag{5.16}$$

$$0 \le x_{v,k} \le 1 \qquad \text{f.e. } v \text{ in } V \text{ and } k \text{ in } F(v), \tag{5.17}$$

$$\textstyle\sum_{k\in F(v)} x_{v,k} = 1 \qquad \text{f.e. } v \text{ in } V, \tag{5.18}$$

$$t_{v,k} \le x_{v,k} L \qquad \text{f.e. } v \text{ in } V \text{ and } k \text{ in } F(v), \tag{5.19}$$

$$c_{v,k} \le x_{v,k} L \qquad \text{f.e. } v \text{ in } V \text{ and } k \text{ in } F(v), \tag{5.20}$$

$$t_{v,k} + p_{v,k} - (1 - x_{v,k})L \le c_{v,k} \qquad \text{f.e. } v \text{ in } V \text{ and } k \text{ in } F(v), \tag{5.21}$$

$$0 \le y_{v,w,k} \le 1 \qquad \text{f.e. } k \text{ in } \{1,\ldots,m\} \text{ and } (v,w) \text{ in } B_k, \tag{5.22}$$

$$y_{v,w,k} + y_{w,v,k} = 1 \qquad \text{f.e. } k \text{ in } \{1,\ldots,m\} \text{ and } (v,w) \text{ in } B_k, \tag{5.23}$$

$$c_{v,k} - (1 - y_{v,w,k})L \le t_{w,k} \qquad \text{f.e. } k \text{ in } \{1,\ldots,m\} \text{ and } (v,w) \text{ in } B_k, \tag{5.24}$$

$$\textstyle\sum_{k\in F(v)} c_{v,k} \le \sum_{k\in F(w)} t_{w,k} \qquad \text{f.e. } (v,w) \text{ in } A, \tag{5.25}$$

where the expression "f.e." means "for each".

We show next that this MILP represents the F-JSS problem. Let (f, Y) be an optimal solution of the F-JSS problem. Let $z := \mathrm{mks}(Y)$. Let $p'_v := p_{v,f(v)}$ for each v. Let s

be the tight schedule for $(V, A \cup Y, p')$. For each v and each k in $F(v)$, let $x_{v,k} = 1$ if and only if $f(v) = k$. For each v and each k in $F(v)$, let $t_{v,k} = s_v$ if $f(v) = k$ and $t_{v,k} = 0$ otherwise. For each v and each k in $F(v)$, let $c_{v,k} = s_v + p_{v,k}$ if $f(v) = k$ and $c_{v,k} = 0$ otherwise. Then z, x, t, and c satisfy constraints (5.14) through (5.21) and well as (5.25). Now define y as follows. For each machine k, let

$$\widetilde{V}_k := \{v \in V : F(v) \ni k \ \wedge \ f(v) \neq k\}$$

and choose a subset C of B_k so that the digraph $(\widetilde{V}_k, C)$ is an acyclic tournament. In addition, let D be the set of all (v, w) in B_k such that $v \in \widetilde{V}_k$ and $f(w) = k$. For each (v, w) in B_k, let $y_{v,w,k} = 1$ if $(v, w) \in Y \cup C \cup D$ and $y_{v,w,k} = 0$ otherwise. This definition of y satisfies constraints (5.22) through (5.24).

Now consider the opposite transformation. Let (z, t, c, x, y) be an optimal solution of the MILP. We shall derive from it a machine assignment f and a good selection Y. For each v, let $f(v)$ be the unique k in $F(v)$ for which $x_{v,k} = 1$; such k exists by virtue of constraints (5.17) and (5.18). Let $p'_v := p_{v,f(v)}$ for each v. For each k, let Y_k be the set of all pairs (v, w) in B_k such that $x_{v,k} = 1$, $x_{w,k} = 1$ and $y_{v,w,k} = 1$. According to (5.23), $Y := Y_1 \cup \cdots \cup Y_m$ is a selection. According to (5.21), (5.24), and (5.25), Y is a good selection (and s defined by $s_v = t_{v,f(v)}$ is a schedule for $(V, A \cup Y, p')$). According to (5.19) through (5.20), $t_{v,k} = c_{v,k} = 0$ unless $k = f(v)$. Finally, due to the minimality of z, constraints (5.14) through (5.16) make sure that $z = \mathrm{mks}(Y)$.

The ÖÖY MILP has at most $7\sum_v |F(v)| + |V| + 3\sum_{k=1}^m |B_k| + |A|$ constraints and at most $3\sum_v |F(v)| + \sum_{k=1}^m |B_k| + 1$ variables. (Hence, the number of constraints is bounded by $7n\gamma + n + 3mn^2 + |A|$ and the number of variables is bounded by $3n\gamma + mn^2 + 1$, where n is the number of operations and $\gamma := \max_v |F(v)|$.)

Analysis. Consider the linear relaxation of the ÖÖY MILP, i.e., drop the requirement that x and y be integer. Now take any instance of the F-JSS problem in which

- $p_{v,k} \leq L/2$ for each v and each k a and
- $|V_k| \geq 2$ for $k = 1, \ldots, m$,

where $V_k := \{v \in V : F(v) \ni k\}$. (We believe many instances of the F-JSS problem do posess these properties.) Then the linear relaxation of the ÖÖY MILP has a solution $(\dot{z}, \dot{t}, \dot{c}, \dot{x}, \dot{y})$ with $\dot{t} = 0$, $\dot{c} = 0$, $\dot{x}_{v,k} = 1/|V_k|$, $\dot{y}_{v,w,k} = 1/2$, and

$$\dot{z} = 0 \,.$$

Hence, the integrality gap of the ÖÖY MILP is unbounded. This seems to be a bad property of this MILP.

5.6 Heuristics for the F-JSS problem

We describe next a few greedy heuristics for the F-JSS problem. Each produces a permutation $(u_1, u_2, \ldots, u_n)$ of V and a corresponding sequence $(f_1, f_2, \ldots, f_n)$ of machines. The permutation is such that $i < j$ whenever $(u_i, u_j) \in A$. The sequence of machines is such that $f_i \in F(u_i)$ for each i. This pair of sequences defines a good selection: just take the set of all ordered pairs (u_i, u_j) such that $i < j$ and $f_i = f_j$.

$n = |V|$

In all our heuristics, each iteration starts with sequences $(u_1, \ldots, u_{q-1})$ and $(f_1, \ldots, f_{q-1})$ such that no arc in A enters the set $U := \{u_1, \ldots, u_{q-1}\}$. Let $A_U := A \cap (U \times U)$, and let Y_U denote the set of all pairs (u_i, u_j) in $U \times U$ such that $i < j$ and $f_i = f_j$. Let s be the tight schedule for $(U, A_U \cup Y_U, p')$, where $p'_{u_i} := p_{u_i, f_i}$. Then the completion time of u_i will be $c_i := s_{u_i} + p'_{u_i}$ and each machine k will become available at time

U

c

avl

$$avl[k] := \max \{c_i : 1 \leq i < q \text{ and } f_i = k\} .$$

(Of course $avl[k] = 0$ when there is no i such that $f_i = k$.) From this information, the heuristics described below choose, each in its own greedy manner, the next operation u_q and next machine f_q.

Heuristic P. The first heuristic (dubbed P, for "past") chooses u_q and f_q in a way that depends only on the past choices $(u_1, \ldots, u_{q-1})$ and $(f_1, \ldots, f_{q-1})$. (The heuristic could also be be called EST, for "earliest starting time.") The basic idea is to choose a pair (u_q, f_q) whose execution can start the earliest. This rule is, in general, satisfied by several pairs. Experience shows that an additional tie-breaking rule can significantly improve the heuristic. In order to describe our tie-breaking rule we must introduce some notation.

Let $\bar{p}$ represent be the mean processing time. Hence, for each operation v,

$$\bar{p}_v = \frac{1}{|F(v)|} \sum_{k \in F(v)} p_{v,k} .$$

Now, for each operation w, let

$$\textsc{Lpf}\,(w, \bar{p}) \tag{5.26}$$

denote the length of a $\bar{p}$-longest path from w in the dag (V, A), i.e., the largest sum of the form $\bar{p}_w + \bar{p}_{z_1} + \bar{p}_{z_2} + \cdots + \bar{p}_{z_l}$ where $(w, z_1, z_2, \ldots, z_l)$ is a (directed) path in (V, A). (Note that this definition of length is slightly different from the one is section 2.1 since the sum includes the term $\bar{p}_{z_l}$.) The tie-breaking rule can now be stated as follow: If there are several candidate pairs (w, k) for the role of (u_q, f_q), choose one that will maximize $\textsc{Lpf}\,(w, \bar{p})$. The details can be seen in figure 5.5. The

heuristic takes time in $O(n|A| + n^2m)$.

```
HEURISTIC-P (n, V, A, F, m, p)
 1  for all k in {1, ..., m} do avl[k] ← 0
 2  U ← ∅
 3  for q ← 1 to n do
 4      for all w in V \ U do
 5          if (v, w) ∈ A for some v in V \ U
 6              then rdy[w] ← ∞
 7              else rdy[w] ← max {c_i : 1 ≤ i < q and (u_i, w) ∈ A}
 8      earlst ← ∞
 9      for all w in V \ U such that rdy[w] < ∞ do
10          for all k in F(w) do
11              t ← max(rdy[w], avl[k])
12              if earlst > t or (earlst = t and mxlpf < LPF(w, p̄))
13                  then earlst ← t
14                       x ← w
15                       l ← k
16                       mxlpf ← LPF(w, p̄)
17      u_q ← x
18      f_q ← l
19      c_q ← earlst + p_{x,l}
20      avl[l] ← c_q
21      U ← U ∪ {u_q}
22  return (u_1, ..., u_n) and (f_1, ..., f_n)
```

Figure 5.5: Heuristic P for the F-JSS problem. At the beginning of line 8, $rdy[w]$ is the ready time for operation w, i.e., the time at which all predecessors of w in the dag (V, A) are complete. (If $rdy[w] = \infty$ then w is not yet ready to be processed.) On line 11, t is the time at which machine k can start processing operation w. At the beginning of line 17, $earlst < \infty$ and x and l are well defined, since the digraph (V, A) is a dag and $U \neq V$. All the "∞" can be replaced by "$\sum_{v \in V} \max_{k \in F(v)} p_{v,k}$". The makespan of the schedule defined by the heuristic is simply $\max_{i=1}^{n} c_i$.

Heuristic F1. Our second heuristic (dubbed F, for "future") chooses u_q and f_q based on the time each machine is expected to spend processing the remaining operations, i.e., the operations in $V \setminus U$.

In order to state the heuristic, we need some notation. Given U and a machine k, let $\textsc{Ert}(U, k)$ be the expected remaining time for machine k, i.e.,

$$\textsc{Ert}(U, k) := \sum_{w} \frac{p_{w,k}}{|F(w)|} , \tag{5.27}$$

where the sum is taken over all w in $V \setminus U$ such that $F(w) \ni k$. The expected makespan relative to $(u_1, \ldots, u_{q-1})$ and $(f_1, \ldots, f_{q-1})$ is then

$$\max_{k=1}^{m} \left(avl[k] + \textsc{Ert}(U, k) \right) . \tag{5.28}$$

Heuristic F1 chooses the next pair (u_q, f_q) so that the expected makespan after vertex u_q is assigned to machine f_q (i.e., the expected makespan relative to $(u_1, \ldots, u_q)$ and $(f_1, \ldots, f_q)$) is as small as possible. The details are given in figure 5.6. The heuristic takes time in $\mathrm{O}(n|A| + n^2m + nm^2)$.

```
HEURISTIC-F1 (n, V, A, F, m, p)
 1  for all k in {1, ..., m} do avl[k] ← 0
 2  U ← ∅
 3  for q ← 1 to n do
 4      for all w in V \ U do
 5          if (v, w) ∈ A for some v in V \ U
 6              then rdy[w] ← ∞
 7              else rdy[w] ← max {c_i : 1 ≤ i < q and (u_i, w) ∈ A}
 8      minemks ← ∞
 9      for all w in V \ U such that rdy[w] < ∞ do
10          for all k in F(w) do
11              emks ← max(rdy[w], avl[k]) + p_{w,k} + ERT(U ∪ {w}, k)
12              for all h in {1, ..., m} \ {k} do
13                  emks ← max(emks, avl[h] + ERT(U ∪ {w}, h))
14              if minemks > emks
15                  then minemks ← emks
16                       x ← w
17                       l ← k
18      u_q ← x
19      f_q ← l
20      c_q ← max(rdy[x], avl[l]) + p_{x,l}
21      avl[l] ← c_q
22      U ← U ∪ {u_q}
23  return (u_1, ..., u_n) and (f_1, ..., f_n)
```

Figure 5.6: Heuristic F1 for the F-JSS problem. At the beginning of line 8, $rdy[w]$ is the ready time for operation w, i.e., the time at which all predecessors of w in the dag (V, A) are complete. At the beginning of line 14, $emks$ is the expected makespan if w is assigned to machine k. Lines 14–17 choose the pair (x, l) that minimizes $emks$. All the "∞" can be replaced by "$\sum_{v \in V} \max_{k \in F(v)} p_{v,k}$". The makespan of the schedule defined by the heuristic is simply $\max_{i=1}^{n} c_i$.

Heuristic PF1. This is an iteration-by-iteration combination of P and F1. See figure 5.7.

HEURISTIC-PF1 (n, V, A, F, m, p)

1 for all k in $\{1, \ldots, m\}$ do $avl[k] \leftarrow 0$
2 $U \leftarrow \emptyset$
3 for $q \leftarrow 1$ to n do
4 for all w in $V \setminus U$ do
5 if $(v, w) \in A$ for some v in $V \setminus U$
6 then $rdy[w] \leftarrow \infty$
7 else $rdy[w] \leftarrow \max\{c_i : 1 \le i < q$ and $(u_i, w) \in A\}$
8 for all w in $V \setminus U$ such that $rdy[w] < \infty$ do
9 for all k in $F(w)$ do
10 $e \leftarrow \max(rdy[w], avl[k]) + p_{w,k} + \text{ERT}(U \cup \{w\}, k)$
11 for all h in $\{1, \ldots, m\} \setminus \{k\}$ do
12 $e \leftarrow \max\big(e,\ avl[h] + \text{ERT}(U \cup \{w\}, h)\big)$
13 $emks[w, k] \leftarrow e$
14 $best \leftarrow \infty$
15 for all w in $V \setminus U$ such that $rdy[w] < \infty$ do
16 for all k in $F(w)$ do
17 $prdct \leftarrow \max(rdy[w], avl[k]) \cdot emks[w, k]$
18 if $best > prdct$ or ($best = prdct$ but $mxlpf < \text{LPF}(w, \bar{p})$)
19 then $best \leftarrow prdct$
20 $mxlpf \leftarrow \text{LPF}(w, \bar{p})$
21 $x \leftarrow w$
22 $l \leftarrow k$
23 $u_q \leftarrow x$
24 $f_q \leftarrow l$
25 $c_q \leftarrow \max(rdy[x], avl[l]) + p_{x,l}$
26 $avl[l] \leftarrow c_q$
27 $U \leftarrow U \cup \{u_q\}$
28 return $(u_1, \ldots, u_n)$ and $(f_1, \ldots, f_n)$

Figure 5.7: Heuristic PF1 is a combination of heuristics P and F1. At the beginning of line 8, $rdy[w]$ is the ready time for operation w, i.e., the time at which all predecessors of w in the dag (V, A) are complete. After line 13, $emks[w, k]$ is the expected makespan if w is assigned to machine k. Lines 14–22 choose the pair (x, l) that will play the role of (u_q, f_q). All the "∞" can be replaced by "$\sum_{v \in V} \max_{k \in F(v)} p_{v,k}$". The makespan of the schedule defined by the heuristic is simply $\max_{i=1}^{n} c_i$.

Heuristic F2. In each iteration, heuristic F2 (a.k.a. "heuristic Cris2") chooses the pair (u_q, f_q) so that $avl[f_q]$ is as small as possible and, subject to this condition, so that the expected makespan after u_q is assigned to f_q is minimized. The heuristic is given in figure 5.8. It takes time in $O(n|A| + n^2m)$.

```
HEURISTIC-F2 (n, V, A, F, m, p)
 1  for all k in {1, ..., m} do avl[k] ← 0
 2  U ← ∅
 3  for q ← 1 to n do
 4      for all w in V \ U do
 5          if (v, w) ∈ A for some v in V \ U
 6              then rdy[w] ← ∞
 7              else rdy[w] ← max {c_i : 1 ≤ i < q and (u_i, w) ∈ A}
 8      minavl ← ∞
 9      for k ← 1 to m do
10          if minavl > avl[k]
11              then if F(w) ∋ k and rdy[w] < ∞ for some w in V \ U
12                       then minavl ← avl[k]
13                            l ← k
14      minemks ← ∞
15      for all w in V \ U such that rdy[w] < ∞ and F(w) ∋ l do
16          emks ← max(rdy[w], avl[l]) + p_{w,l} + ERT(U ∪ {w}, l)
17          for all h in {1, ..., m} \ {l} do
18              emks ← max (emks, avl[h] + ERT(U ∪ {w}, h))
19          if minemks > emks
20              then minemks ← emks
21                   x ← w
22      u_q ← x
23      f_q ← l
24      c_q ← max(rdy[x], avl[l]) + p_{x,l}
25      avl[l] ← c_q
26      U ← U ∪ {u_q}
27  return (u_1, ..., u_n) and (f_1, ..., f_n)
```

Figure 5.8: Heuristic F2 (a.k.a. Cris2) for the F-JSS problem. At the beginning of line 8, $rdy[w]$ is the ready time for operation w, i.e., the time at which all predecessors of w in the dag (V, A) are complete. Lines 8–13 choose the machine l that will play the role of f_q. At the beginning of line 19, $emks$ is the expected makespan if w is assigned to machine l. Lines 14–21 choose operation x that will minimize $emks$. All the "∞" can be replaced by "$\sum_{v \in V} \max_{k \in F(v)} p_{v,k}$". The makespan of the schedule defined by the heuristic is simply $\max_{i=1}^{n} c_i$.

Heuristic F3. The next heuristic is a simplified version of F2 that produces surprisingly good results on our tests. The expected total time (ett) on machine k relative to $(u_1, \ldots, u_{q-1})$ and $(f_1, \ldots, f_{q-1})$ is

$$avl[k] + \textsc{Ert}(U, k)\ . \tag{5.29}$$

In each iteration, heuristic F3 chooses the pair (u_q, f_q) so that $avl[f_q]$ is as small as possible and, subject to this condition, so that the ett on machine f_q relative to $(u_1, \ldots, u_q)$ and $(f_1, \ldots, f_q)$ will be as small as possible. The heuristic is given in figure 5.9. It takes time in $\mathrm{O}(m + n^2 + n|A|)$.

HEURISTIC-F3 (n, V, A, F, m, p)

```
 1  for all k in {1,...,m} do avl[k] ← 0
 2  U ← ∅
 3  for q ← 1 to n do
 4      for all w in V \ U do
 5          if (v, w) ∈ A for some v in V \ U
 6              then rdy[w] ← ∞
 7              else rdy[w] ← max {c_i : 1 ≤ i < q and (u_i, w) ∈ A}
 8      minavl ← ∞
 9      for k ← 1 to m do
10          if minavl > avl[k]
11              then if F(w) ∋ k and rdy[w] < ∞ for some w in V \ U
12                       then minavl ← avl[k]
13                            l ← k
14      minett ← ∞
15      for all w in V \ U such that rdy[w] < ∞ and F(w) ∋ l do
16          ett ← max(rdy[w], avl[l]) + p_{w,l} + ERT(U ∪ {w}, l)
17          if minett > ett
18              then minett ← ett
19                   x ← w
20      u_q ← x
21      f_q ← l
22      c_q ← max(rdy[x], avl[l]) + p_{x,l}
23      avl[l] ← c_q
24      U ← U ∪ {u_q}
25  return (u_1,...,u_n) and (f_1,...,f_n)
```

Figure 5.9: Heuristic F3 for the F-JSS problem. At the beginning of line 8, $rdy[w]$ is the ready time for operation w, i.e., the time at which all predecessors of w in the dag (V, A) are complete. Lines 8–13 choose the machine l that will play the role of f_q. At the beginning of line 17, ett is the expected total time for machine l assuming that w is assigned to machine l. Lines 14–19 choose operation x that minimizes ett. All the "∞" can be replaced by "$\sum_{v \in V} \max_{k \in F(v)} p_{v,k}$". The makespan of the schedule defined by the heuristic is simply $\max_{i=1}^{n} c_i$.

Test results. Heuristics P and F1 were applied to the set of test instances mentioned in figure 5.1. The results are summarized in figure 5.10.

instance	P	F1	PF1	F2	F3
SFJS01	66	86	66	86	86
SFJS02	107	128	107	107	107
SFJS03	255	236	255	314	304
SFJS04	367	409	409	417	409
SFJS05	143	144	145	144	144
SFJS06	360	357	360	390	330
SFJS07	407	507	397	397	457
SFJS08	273	296	273	273	340
SFJS09	230	247	230	215	220
SFJS10	608	914	617	724	693
MFJS01	526	639	522	832	576
MFJS02	540	636	570	790	587
MFJS03	655	719	632	740	614
MFJS04	690	847	768	827	684
MFJS05	690	708	733	778	644
MFJS06	838	953	821	1239	1063
MFJS07	1130	1531	1064	1773	1241
MFJS08	1129	1248	1137	1447	1371
MFJS09	1343	1508	1339	1824	1525
MFJS10	1559	1845	1500	1970	1642
MK01	49	69	47	76	48
MK02	41	44	46	48	35
MK03	204	287	204	398	235
MK04	73	103	69	119	100
MK05	186	232	182	246	218
MK06	98	151	99	168	91
MK07	214	207	193	239	200
MK08	523	795	523	716	602
MK09	336	544	329	552	405
MK10	274	455	254	488	285
MK11	698	752	641	858	700
MK12	566	744	550	599	571
MK13	500	741	470	771	486
MK14	719	1349	694	920	777
MK15	443	611	394	798	512

Figure 5.10: Applying heuristics P, F1, PF1, F2, and F3 to the test instances mentioned in figure 5.1. The table records the makespan of the schedules computed by the various heuristics. Somewhat surprisingly, F3 is often better than F1 and F2.

Bibliography

[AC91] D. Applegate and W. Cook. A computational study of the job-shop scheduling problem. *ORSA Journal on Computing*, 3:149–156, 1991. 9, 10

[BDP96] J. Błażewicz, W. Domschke, and E. Pesch. The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research*, 93(1):1–33, 1996. 9

[Bra93] P. Brandimarte. Routing and scheduling in a flexible job-shop by tabu search. *Annals of Operations Research*, 41:157–183, 1993. Internet: http://www.springerlink.com/index/10.1007/BF02023073. 17

[Car82] J. Carlier. The one machine sequencing problem. *European Journal of Operations Research*, 11:42–47, 1982. 11, 12

[CPL12] IBM ILOG. *CPLEX Optimizer*, 12.1 edition, 2012. Internet: http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/. 18

[FMJ07] P. Fattahi, M.S. Mehrabad, and F. Jolai. Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. *Journal of Intelligent Manufacturing*, 18:331–342, 2007. 17, 21

[Mas] M. Mastrolilli. Flexible job shop problem. Internet: http://www.idsia.ch/~monaldo/fjsp.html. 17

[RDL+09] S. Rai, C.B. Duke, V. Lowe, C. Quan-Trotter, and T. Scheermesser. LDP lean document production—O.R.-enhanced productivity improvements for the printing industry. *Interfaces*, 39(1):69–90, 2009.

[Sch71] L.E. Schrage. Obtaining optimal solutions to resource constrained network scheduling problems. In *AIIE Systems Engineering Conference*. American Institute of Industrial Engineers, 1971. 11

[ZJL+10] J. Zeng, S. Jackson, I-J. Lin, M. Gustafson, E. Hoarau, and R. Mitchell. On-demand digital print operations: A simulation based case study.

Technical report, Hewlett Packard, Palo Alto, CA, September 2010. Confidential. 2

[ÖÖrY10] C. Özgüven, L. Özbakır, and Y. Yavuz. Mathematical models for job-shop scheduling problems with routing and process plan flexibility. *Applied Mathematical Modelling*, 34:1539–1548, 2010. 21

Index