Dispatching Work: Finding the best dispatching method for real job-shops

by

Andrew A.H. Brown

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Management Sciences

Waterloo, Ontario, Canada, 2014

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Motivated by a situation observed by our industry partner, we test if changing dispatch methods within a job-shop can reduce the percentage of late jobs while not reducing the maximum lateness across all jobs, the two key performance indicators (KPIs) of interest. Using data provided by our industry partner, we show that the earliest operation due date (EODD) dispatch rule is the best rule for them. In addition, we propose an alternative idea for random job shop data, the routing distribution, and we compare dispatching rules performance using KPI frontiers under different routing distributions. We find that EODD is one of several dispatching rule which consistently lie on the KPI frontier for different job routing distributions. We further show that using multiple dispatch rules across several job-shop departments does improve a job-shop's performance on the KPIs, though the improvement is small. Lastly, we leave the readers with some insight into determining which dispatch rules should be considered for different job-shops.

Acknowledgments

I would like to thank my advisors Stan Dimitrov and Ada Barlatt for their enthusiasm, guidance and willingness to let me share in this research problem with me. I also thank my Master's committee for their many useful suggestions for improving this thesis. I am also very grateful to our industry partner for their generous access and assistance. I thank the Department of Management Science for the academic opportunities afforded me as a graduate student.

Contents

List of Figures

1. Introduction

Manufacturing is an important part of Canada's future economic development, representing 14% of Canada's Gross Domestic Product, but it is facing many new challenges from around the globe. To address those challenges, Canadian manufacturers need to become more efficient (CMC, 2012). Many companies, like our industry partner, are seeking efficiency gains by implementing Industrial Engineering methods in their job-shops, particularly for their production scheduling. In this thesis, we show that our industry partner's current scheduling method is a fine choice, that multiple rule methods can increase the efficiency of their production scheduling further, and that, for the most part, these results generalize to other job-shops.

We were tasked by our industry partner to propose new scheduling methods for their job-shop to reduce the number of late jobs. In observing their schedule, we determined that any scheduling method we recommend should be *human implementable*; a method is human implementable if it usable by schedulers with minimal training using existing scheduling equipment such as work-in-progress lists, and job boards. Our industry partner uses a human implementable method currently, so a new human implementable scheduling method can be implemented by replacing the current method, with little change to other business processes. Indeed, an important implication of human implementability is that none of our recommended methods can require new software, equipment acquisitions, substantial retraining, or major corporate restructuring.

Dispatching rules are a commonly used scheduling method, and most are human implementable. To define them, we first define a *job-shop*. As in many production models, a job-shop has a number of machines or resources on which the production work is processed. The production in job-shops is organized into *jobs*: an ordered list of operations each completed using a specific machine or resource for a set amount of time. Each job has a *due date* by which time the job is ideally complete. At each machine, there are enqueued jobs waiting to be processed, and a *dispatching rule* is a scheduling method used every time a machine finishes processing an operation, with the function to to choose the next enqueued job to 'dispatch' for processing.

Dispatching rules are typically quite simple:

> All else being equal, process the job with the earliest due date.

The earliest due date (EDD) rule, as this example is known, is human implementable; a person can determine the next job to select as per EDD with only a list of enqueued jobs and their due dates. Our industry partner currently uses a variant of EDD called earliest operation due date (EODD) defined in Section 3.5.2. There are many other well-known dispatching rules; Pinedo (2009) defines several. In this thesis, we consider only those dispatching rules which can be assessed using a list to ensure human implementability. We show by simulating our industry partner's factory that EODD minimizes the number of late jobs

while completing all jobs, whereas another rule, shortest operation processing time (SOPT) first, defined in Section 3.5.6, reduces the number of late jobs by as much as 50% over EODD though a small set of jobs is left incomplete.

In addition, we show that multiple rule scheduling methods, methods where two or more dispatching rules are assigned to groups of machines, are slightly more effective. Our industry partner organizes their machines and production scheduling using departments, and they have experimented with several ways of setting those departments, grouping machines by function for instance. We propose new scheduling methods which assign different dispatching rules to the groups our industry partner used. We show that the best performing of these multiple rule methods out-perform using single rule, though typically by less than 1% fewer late jobs.

We then generalize the above work by simulating the functioning of all our scheduling methods on randomly created production data. That production data is created using several empirical distributions based on our industry partner's data, as well as five different routing distributions. A *routing distribution* is a set of discrete probability distributions, each on the set of job-shop machines, with a distribution assigned to or generated for each operation of every job. Each operation's probability distribution is sampled to determine the machine processing that operation of every job. A simple example of a routing distribution is what we have called a uniform routing distribution, previously called an open job-shop (Philipoom and Fry, 1990), in which every probability distribution for every operation is a uniform distribution on the set of machines.

We suspect that the uniform routing distribution is commonly used due to its simplicity, but production data generated using uniform routing is a poor model for our industry partner's production data. Overall, the performances of our dispatching rules is very different on the uniformly generated production data compared to the overall performance on our industry partner's data. One difference between them is the key-performance-indicator (*KPI*) Pareto frontiers or *KPI frontiers*. The KPI frontier for the industry partner data exhibits a clear trade-off between the two indicators, percentage of late jobs and maximum lateness; whereas, the performances of each dispatching rule on the KPI frontier for the uniformly generated data are nearly identical, with less than 1% performance difference in both KPIs. That difference, and others, are due to the specialization of some or all of the machines in a real job-shop; for example some machines process the leading operations, while others process the trailing operations.

In an effort to better model that specialization, we define, in Section 3.3, four routing distributions beyond the uniform routing distribution. We compare routing distributions by calculating the KPI frontier of the best performing dispatching rules on each routing distribution. We find that the performance of the best dispatching rules varies substantially, more than we initially expected. However, for most of the routing distributions considered, EODD and SOPT are still, on average, good dispatching rules.

Now, there are many scheduling heuristics and algorithms studied in both OR and artificial intelligence with significantly better theoretical performance than dispatching rules. Indeed, our industry partner has previously scheduled their production with one algorithm implemented by a computerized scheduling system. However, in McKay et al. (1988), the authors find that the theoretical methods studied are likely irrelevant to real job-shops, as they do not account for many of the realities faced by schedulers in real job-shops. The scheduling methods we describe here, both single rule and multiple rule methods, avoid being irrelevant in the following ways.

Firstly, most job-shop algorithms require extensive computations, so the reasons for and interactions among the choices made by these algorithms can be quite opaque. When changes are required, that opacity makes altering portions of a created schedule difficult. In contrast, the reasons for the choices each dispatching rule makes are straight forward: calculate the earliest due date, so that jobs do not go past due. With that clarity, schedulers can easily weigh the importance of a dispatching rule choice against other manufacturing requirements such as the need to conduct preventative maintenance.

Secondly, job-shop algorithms typically create complete schedules which specify the start and stop times of all operations for all jobs in progress. However, the situation on a shop floor changes very quickly with new orders, machine break downs and many other considerations, so the later half of a complete schedule is typically changed dramatically, a computational waste. Job-shops algorithms cannot capture enough of the complexity and instability inherent in scheduling a real job-shop (McKay et al., 1988). The scheduling methods proposed here avoid that waste, as dispatching rules make scheduling choices in real time.

Finally, neither dispatching rules nor multiple rule methods replace or reduce the scope of job-shop employees in the scheduling process, whereas computerized OR-solutions often do replace or reduce the scope of employees. For instance, it is difficult for anyone to criticize a computer generated schedule, as any suggested changes could require abandoning large portions of the schedule due to the cascading effect of changes in job-shops. Hence, employee criticism and suggested changes can go unheeded. After being unheeded often enough, employees stop offering input, acquiescing to the schedule regardless of their better judgment. While using their computer scheduling system, our industry partner experienced that employee acquiescence, and they found that employee involvement in many other aspects of production declined, negatively affecting production as a result (Industry-Partner, 2013). The scheduling methods here, on the other hand, allow employees, informed by the changing situation, to easily make, contribute to, or criticize scheduling choices. In particular, the suggestions of any of our scheduling methods can be easily over ruled in favour of other priorities.

In Section 3.5, we define several dispatching rules for use in our simulations either alone or in combinations. We conduct a tournament using a job-shop simulation, described in Section 3.1, and compare the performance of each sim-

ulation using two of our industry partner's KPIs: the percentage of late jobs and the maximum lateness. The simulations we seek are on the KPI frontier for these two KPIs. Our simulations are run on data provided by our industry partner, as well as random production data generated using empirical distributions based on that same data and the routing distributions defined in Section 3.3. We present the results of our simulations in Sections 4 and 5. We conclude, in Section 7.1, with a specific recommendation for both our industry partner and for job-shops in general.

2. Related Work

The study of job-shop problems has proceeded similarly to other NP-hard problems. First, the problem is formulated and optimal algorithms are proposed (Manne, 1960). Later, after an NP-hardness proof is found, by Garey et al. (1976) for job-shops, researchers assume that all optimal algorithms for the problem have prohibitively long run-times, and researchers began developing approximation algorithms and heuristic solutions in favour of optimal algorithms. Approximation algorithms and heuristic solutions find good, though not optimal, solutions to hard problems in polynomial time, and the polynomial run time allows them to be more easily employed by industry. Dispatching rules, a family of heuristics for job-shops and the focus of this thesis, were proposed by Panwalkar and Iskander (1977).

There are a variety of algorithms and heuristics, which in theoretical terms, out perform dispatching rules such as the approximation algorithms (Shmoys et al., 1994), the shifting bottle neck algorithm (Adams et al., 1988), or constraint programming methods (Nuijten and Aarts, 1996) to name just a few. However, the solution of many of these algorithms and heuristics are complete schedules, and as was mentioned above, McKay et al. (1988) found that, among other things, actual job-shops are prone to rapid and unpredictable change. Complete schedules implicitly assume that job-shops are more stable than is really true. Dispatching rules avoid this problem by not creating complete schedules; they merely specify what should be worked on next by a free machine or resource.

Another, more recent heuristic approach is to search through a job-shop instance's solution space to find solutions with good objective values. There is a large variety of such search algorithms: ant colony algorithms (Colorni and Dorigo, 1994; Huang and Liao, 2008); differential evolution algorithms (Pan et al., 2008; Wei-ling and Jing, 2013); genetic algorithms (Yu and Liang, 2001; De Giovanni and Pezzella, 2010); local-search (Vaessens et al., 1994; Vela et al., 2008); particle swarm algorithms (Sha and Hsu, 2006; Zhang and Wu, 2010), simulated annealing (Tavakkoli-Moghaddam et al., 2008; Zhang and Wu, 2011), and tabu-search algorithms (Nowicki and Smutnicki, 2005; Armentano and Scrich, 2000). These algorithms also create complete solutions, but there is another problem, that of understanding generated schedules.

Search algorithms all use various steps with no clear analogue to scheduling a real job-shop, such as randomly generated schedules, "scent trails" along a "path" of schedules or "moves" from schedules to other schedules. The lack of analogy for these steps makes it difficult for industry schedulers to understand the choices and assumptions made by the search algorithm while generating a schedule, so it is difficult to know how to alter a schedule and to what degree when the real situation changes. The proactive scheduling algorithm proposed in Goren et al. (2012), while avoiding the problems of complete schedules, shares the analogy problem because it employs tabu-search. The dispatching rules we consider here, namely human implementable dispatching rules, are specifically

chosen or designed to have clear analogies to real job-shop scheduling, involving quick calculations or lookups, such as finding the earliest due date amongst a set of jobs.

We consider dispatching rules from several sources (Panwalkar and Iskander, 1977; Pinedo, 2009), as well as some new rules. We compare their performance in job-shop simulation using the percentage of late jobs and the maximum amount a job is completed past due. Similar analyses of dispatching rule performance on job-shop simulations can be found in Kaban et al. (2012), or Sculli and Tsang (1990). Our goal here is slightly different from those papers in that we not simply looking for the best performing rule under our chosen KPI or KPIs. Instead, we are recommending changes to our how industry partner's schedules to reduce the number of their late jobs, so we sought rules which outperform their current dispatching rule, earliest operation due date.

We also take up a study of what we call routing distributions, defined in Section 1, as they appear to be an under studied aspect of job-shop. Some job-shop researchers explicitly state the routing distribution used to create their random production data as in Adams et al. (1988), Shah (2004), or Ruiz and Vázquez-Rodríguez (2010), but being explicit about this does not seem to be common among researchers. Moreover, different routing distributions are known to affect the performance of dispatching rules (Philipoom and Fry, 1990).

As mentioned above, a job set has a uniform routing distribution, also known as an open (Philipoom and Fry, 1990) or a pure job-shop (Holstein and Berry, 1970), if for a given job length, every route through the job-shop's machines is equally probable. Uniform routing implies that it is unlikely there will be any routing "features" like bottlenecks or favoured routes in the job set. Any route is equally as likely as any other. However, the routing distribution observed in our industry partner's production data included several features such as loose staging and bottle necks, among other things, suggesting that uniformly distributed routing is a poor model of industry demand. We compare our scheduling methods on random production data created from several different, explicitly stated, routing distributions, including uniform routing. We show that our set of dispatching rules collectively perform quite differently on each routing distribution, so routing distributions are an important consideration in random production data creation.

The last new aspect of this thesis is the focus on human implementable dispatching rules. To our knowledge, the studies of dispatching rules in the literature have not previously sorted the rules upon this criteria. The focus on human implementability comes from the need to make improvement recommendations to our industry partner that easily fit into their existing scheduling processes. In personal communication with our industry partner, we learned that they use an effective, human-centric scheduling process which has improved performance on all of their key performance indicators, reversing the production decreases attributed to a previously used computerized scheduling system (Industry-Partner, 2013). Their current process employs a human implemented dispatching rule,

earliest operation due date first, so a different rule is conceivably an easy change to make, and one that maintains the centrality of the human scheduler along with the attendant benefits our industry partner experiences.

3. Notation and Methods

In this section, we introduce the necessary definitions and notation to describe a job-shop instance, our simulations, and the particulars of our solutions. We begin with the definition of our job-shop simulations.

3.1. Job-Shop Simulation

We model job-shops using a computer simulation. The setup of our simulations consists of

1. a job set $\mathcal{J}$, Section 3.3, and
2. scheduling method, Section 3.5.

A job-shop is simulated using a discrete time line implemented using SimPy to track factory events: job releases, processing starts, processing completions, and job transits between machines (Lunsdorf et al., 2012). The simulations are deterministic because all of the variables specified in the job set are constants, and all the scheduling methods are deterministic. The code itself is shown in Appendix B.

Throughout a simulation, the current state of each machine and each released job is available for dispatching rules. The current state of a released job is given by a combination of the amount of processing completed on j, and the location of j, whether in transit between machines or at a specific machine. In Figure 1, job 0 is at machine m, 2 of 5 operations of job 0 in the figure are complete, and the third is 43% complete. The state of a machine m includes two pieces of information. The first is the queue, q_m, of all the jobs waiting at m. The second piece is either that m has completed p percent of operation i of job j or that m is in sleep mode waiting for a new job to enter the job queue, which for m is denoted q_m. In Figure 1, the state of m is the list of jobs 0 through 4, and that operation 3 of job 0 is 43% complete.

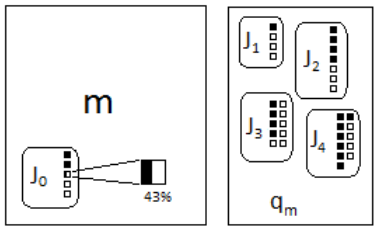

Figure 1: The state of a machine m; jobs 1 to 4 are enqueued and the third operation of job 0 is being processed on m.

The dispatching rule for a machine m is triggered whenever m finishes processing an operation. The rule chooses a job enqueued at m, and m begins processing it. If the rule finds that q_m is empty, $q_m = \emptyset$, then m is put to "sleep" until a new job enters q_m. From every simulation, the history of the simulation is returned as a list of all the start times of all operations. We calculate the two KPIs for each simulation: percentage of late jobs, and maximum a job completed past due.

3.2. Job-shop Demand Definitions and Notation

Figure 2 is an example of a job used in our simulation. Each job has several parameters: job name, release date and due date, and an ordered list of operations. An *operation* is a task which must be processed on a particular machine for the number of hours given by the *processing time* , and the operation *index* is the position of the operation in the list of operations. The first operation in Figure 2 has index 1 and takes 4.4 hours on machine m54. The *release date* is the date after which processing the first operation can begin, 10/18/13 in the figure. A *job set* consists of a number of such jobs.

Name	**Release Date**	**Due Date**
j1774	10/18/13 0:01	01/10/14 23:59

Index	**Machine**	**Processing (hrs)**
1	m54	4.40
2	m57	0.24
3	m03	0.24
4	SUB	105.0
5	m07	1.24
6	m02	5.68
7	SUB	45.0
8	m09	0.10

Figure 2: An example of a job.

The job in Figure 2 also has two special operations each processed on a machine called SUB. This machine stands for processing which is subcontracted, such as heat treatment in metal work. The processing time includes the transportation handling and actual processing by the contractor. Our industry partner has several contractors available for each type of subcontracted processing, and we model this in our simulations by treating SUB as the only machine which can process multiple operations at the same time.

The mathematical notation required to describe the above is as follows. A job set is denoted $\mathcal{J} = \{j_1, j_2, \ldots, j_N\}$, and the set of all operations in $\mathcal{J}$ is $Op(\mathcal{J})$. As described above, each job $j \in \mathcal{J}$ has

- an ordered list of operations $O_j = (o_1, o_2, \ldots, o_\ell)$,

- a release date r_j, and
- a due date d_j.

The ith operation of job j is be denoted o_{ij}. Each operation has

- an assigned machine m_{ij} on which o_{ij} is processed, and
- a processing time $p_{ij} > 0$ of o_{ij}.

Our notation here differs from the standard notation set in Pinedo (2009) where operations are denoted as the tuple (m_{ij}, j). We found our exposition aided by the above notation.

We also make some assumptions based on details of our industry partner's job-shop (Industry-Partner, 2013).

- We assume an operation cannot be preempted, so once processing begins on an operation, it cannot be stopped until complete.
- Operations are completed in a linear order by index.
- Release dates are not required to be the same for all jobs.
- Each job can be processed on one and only one machine at a time.
- Each machine can process one and only one operation at a time.
- Setup times are included in the processing time, and they are not sequence dependent.
- Re-entrant jobs are permitted, that is two or more operations of a job may be processed on the same machine, but re-entrance can only occur after a SUB operation.
- One day is required to transport a job between different machines.

Another assumption we make is that the simulated machines do not break down or need maintenance. It is clear that in a real job-shop machines do break down and do require regular maintenance. To account for this lack of down time, our simulations use the same number of hours of processing per day which our industry partner uses for their planning, 80% of the true capacity of each machine. Because of these differences and others, we cannot make a fair comparison between performances in the simulation to the actual performance of our industry partner. However, all the simulations share the same advantages or disadvantages, so simulated performances of scheduling methods can be compared. Hence, the simulation of our industry partner's scheduling method, EODD, can be compared to the simulated performances of all the other scheduling methods.

3.3. Random Job sets

Job sets represent the work manufacturers complete for clients, and these sets are the principle input of our job-shop simulator. We are fortunate enough to

have access to two years worth of production data from our industry partner, representing several thousand jobs; that data was formed into a job set, which we denote as $\mathcal{J}_{Ind}$ throughout. We wished to apply our results to job-shops other than our industry partner's, so we created additional job sets using several probability distributions derived from our industry partner's production data. The creation of a random job set $\mathcal{J}$ proceeds according to Algorithm 1. We note several things about the algorithm.

First, all but one of the parameters needed to create operations, jobs and thus job sets are sampled from our industry partner's data, so operation processing times, job lengths, job due dates etc are all sampled from $\mathcal{J}_{Ind}$. Indeed, we construct each job set so that they each have the same number of machines as $\mathcal{J}_{Ind}$, and within 0.05% of the same number of operations in $Op(\mathcal{J}_{Ind})$. The routing distribution is the one parameter which we do not necessarily sample from $\mathcal{J}_{Ind}$; instead, we use one of the routing functions defined below to determine the job routings.

Second, the machine SUB, representing subcontracted work, appears in every random job set. The steps 7 to 10 of Algorithm 1 show how an operation is assigned to SUB. Our industry partner uses subcontracting at specific points in the processing of a job, points which depend on the length of the job. We model that specificity by using the subset of operations described in step 7: all ith operations of jobs with the same number of operations as j. Recall from Section 3.2 that a machine may appear multiple times for a job only if the occurrences are separated by a SUB operation. This re-entrance condition is enforced by steps 14 and 15 of the algorithm, not by the routing function.

Lastly, Algorithm 1 is used to create all of the random job sets. The algorithm takes a single parameter, the routing function denoted RF. Each routing function takes four parameters: the set of machines $\mathcal{M}$ needed to process $\mathcal{J}_{Ind}$, the job j, the current operation index i, and the current job set $\mathcal{J}$

$$\mathrm{RF}(\mathcal{M}, j, i, \mathcal{J}).$$

Each call to RF returns a discrete probability distribution over $\mathcal{M}$, the set of machines. The machine distribution is sampled at step 13 of Algorithm 1 to determine the machine for the next operation of a particular job. In this section, we define the five different routing functions which implement the five routing distributions: Uniform defined in Section 3.3.1, Empirical in Section 3.3.5, Triangle Section 3.3.2, Preferential Attachment Section 3.3.3, and Preferential Flow in Section 3.3.4. The main difference between them is how the machine distributions are defined. For instance, the uniform routing function always returns a uniform probability distribution over all machine for each operation, whereas the triangle routing function returns a triangle probability distribution. We find that dispatching rule performance varies a great deal depending on the different routing distributions implemented by these functions, so dispatching rule performance is quite sensitive to the routing distributions. The remainder

of this section defines the five routing functions used to implement our routing distributions.

Algorithm 1 Job Set Creation

Require: A routing function:

$$\mathrm{RF}(\mathcal{M}, j, i, \mathcal{J}).$$

Require: The industry partner job set $\mathcal{J}_{Ind}$.

1: **procedure** BUILD JOB SET
2: **while** $|Op(\mathcal{J})| < |Op(\mathcal{J}_{Ind})|$ **do**
3: Sample the release date r_j from $\mathcal{J}_{Ind}$.
4: Sample the difference $d_j - r_j$ from $\mathcal{J}_{Ind}$.
5: Sample the number of operations $|O_j|$ from $\mathcal{J}_{Ind}$.
6: **for** i from 1 to $|O_j|$ **do**
7: Sample o from the ith operations of $|O_j|$ length jobs in $\mathcal{J}_{Ind}$.
8: **if** o is processed on SUB **then**
9: m_{ij} = SUB.
10: Sample processing time p_{ij} from all SUB operations in $\mathcal{J}_{Ind}$.
11: **else**
12: Sample p_{ij} from all non-SUB operations in $\mathcal{J}_{Ind}$.
13: Sample a machine m_{ij} from the machine distribution.

$$\mathrm{RF}(\mathcal{M}, j, i, \mathcal{J}).$$

14: **if** m_{ij} is a repeat since last SUB **then**
15: Return to step 7.
16: Add o_{ij} to job j
17: Add j to job set $\mathcal{J}$.
18: **return** $\mathcal{J}$

3.3.1. Uniform Routing

The uniform routing function returns a discrete, uniform probability distribution over $\mathcal{M}$, so the probability of sampling any machine $m \in \mathcal{M}$ is

$$P(m) = \frac{1}{|\mathcal{M}|}.$$

This routing function can return the same machine on consecutive samples, but the constraint noted above and enforced during Algorithm 1 only allows repeated machines if every pair of instances are separated by the machine SUB. The returned uniform distribution does not depend on the operation number i, the job itself or the current job set $\mathcal{J}$. A job set is called a uniform job set or is said to be uniformly routed if it was created by Algorithm 1 using

the uniform routing function. These job-sets have also been called open job-shops (Philipoom and Fry, 1990).

The first job sets we simulated after $\mathcal{J}_{Ind}$ were uniform job sets. We noticed that the performance of all dispatching rules was significantly better than we observed when simulating $\mathcal{J}_{Ind}$. The uniform job sets seemed "easier" for our dispatching rules to process. The following routing functions are all attempts to find "harder" job sets, where the collective performance of our dispatching rules better resembles their collective performance on $\mathcal{J}_{Ind}$.

3.3.2. Triangle Routing

One of the features observed in $\mathcal{J}_{Ind}$ is that the total number of operations assigned to each machine varies quite widely. An example of how the totals can vary in $\mathcal{J}_{Ind}$ is shown in the left panel of Figure 3. For uniform routing, the total number of operations processed by each machine is roughly the same, as we see in the middle panel of Figure 3. The triangle routing function is one attempt to create job sets with a wide variance among the processing totals, similar to what we see in the left panel of the figure. We call a job set created with a triangle routing function a triangle job set, or that it is a triangularly routed job set.

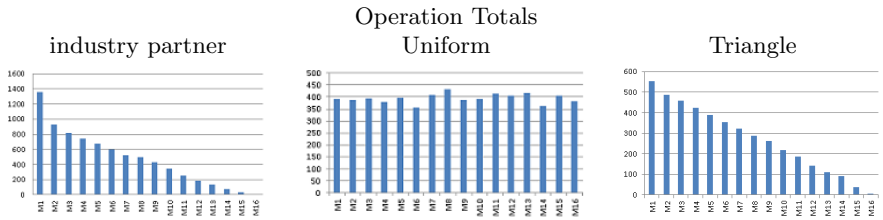

Figure 3: The left panel shows how the total number of operations processed at each machine in $\mathcal{J}_{Ind}$ can vary. The middle panels shows that the totals do not statistically vary for uniform routing. The right panel shows that the total number of operations processed per machine in a triangle job set are subjectively similar to the totals for $\mathcal{J}_{Ind}$.

This routing distribution is implemented using a continuous triangle distribution, a probability distribution with a triangle shaped density graph, as in Figure 4. The parameters for this distribution are the range $[a, b]$ with non-zero density and the mode c, which is the position of the peak in Figure 4. The discrete triangle distribution is obtained by taking the floor of any sample from the continuous triangle distribution. Hence, the probability of sampling the kth machine, m_k in $\mathcal{M}$ is

$$P(m_k) = P(k \leq T < k + 1)$$

where T is a random sample of a triangle distribution.

One sees that those machines "near" the peak of the triangle will appear more often in $\mathcal{J}$, so those machines will process more operations than other machines.

In the right most panel of Figure 3, there is indeed a wide variance in the total number of operations processed on each machine, which is the aim of the triangle routing distribution.

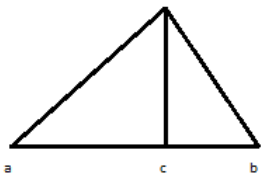

Figure 4: A triangle probability distribution with density on the range $[a, b]$, and mode c

3.3.3. Preferential Attachment Routing

The triangle routing function creates job sets with a lot of variance in the total number of operations processed on each machine, which is similar to $\mathcal{J}_{Ind}$. However, looking at the histograms in Figure 5, we note a difference between $\mathcal{J}_{Ind}$ and a triangularly routed job set. The bar in each range in the left panel of Figure 5 counts the number of machines in $\mathcal{J}_{Ind}$ which process a number of operations between the bounds of that bar; for example, the number of machines which process between 400 and 600 operations is nine. The middle panel is similar, but the histogram is from a triangular job set. Roughly the same number of machines fall into each category in the middle panel, whereas the left panel shows a great deal of variance between the categories. A few machines in $\mathcal{J}_{Ind}$ process a significant proportion of all the operations while most machines process comparatively few operations.

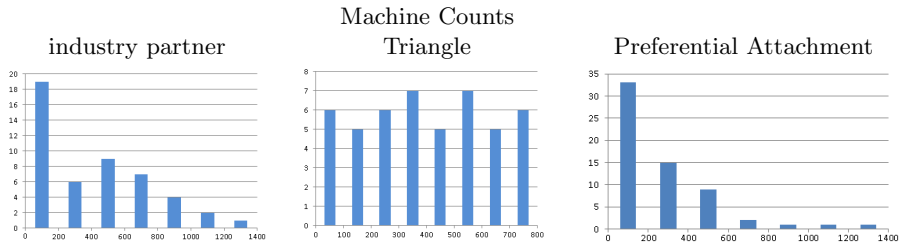

Figure 5: The left panel shows a histogram counting the number machines from $\mathcal{J}_{Ind}$ which process a number of operations processed between the bounds of each bar. For example, 9 machines process between 400 and 600 operations. The middle and right panels show the same counts for triangular job sets and for preferential attachment job sets.

The relationship observed for $\mathcal{J}_{Ind}$ in Figure 5 is similar to a power law relation, which is the same relationship one observes in network formation models when

the formation is governed by preferential attachment. Preferential attachment is an idea introduced by Barabási and Albert (1999), and it is now a cornerstone idea of complex systems theory. It deals with the evolution of networks, both physical and abstract, and states that the more links a node in a network has the more likely it is to form additional links in the future. There is a preference for new links to be added to highly connected nodes. One of the principle features of networks which evolve by preferential attachment is that there are small number of nodes with a large number of links, while all the other nodes have comparatively very few links.

Job-shops can be viewed as networks where the machine are nodes, and consecutive steps within a job form the links. In this view, a smaller number of machines in $\mathcal{J}_{Ind}$ are highly connected, and a rest of the machines have relatively few connections. We propose a preferential attachment model among the machines as an improved model for $\mathcal{J}_{Ind}$ over both uniform routing and triangular routing, as a preferential attachment job set will have a similar variance of the number of connections. The right panel in Figure 5 shows the machine counts for the preferential attachment job sets, and the histogram for it and the industry partner job set exhibit similar patterns of those counts: a large number of machine processing few operations and a small number of machines processing many operations. Job sets created with this routing function are called preferential attachment job sets.

The preferential attachment routing function returns a probability distribution from which it is more likely to sample a highly connected node, that is a machine which processes a large number of operations. However, every sampling adds another machine to $\mathcal{J}$, the job set being created, so the probabilities of choosing a given machine are updated after each sample. In particular, the probability of sampling a machine m increases after each time m is added to a job in $\mathcal{J}$, so this routing function uses the fact that Algorithm 1 creates $\mathcal{J}$ operation by operation.

The probability of sampling any m is calculated as follows. Consider all the operations currently added to $\mathcal{J}$. For each machine m, count the number of times an operation has been assigned to be processed on m, and denote that count C_m. The probability of sampling m from the distribution returned by preferential routing function is

$$P(m) = \frac{1 + a_d C_m}{\sum_{\ell \in \mathcal{M}} 1 + a_d C_\ell}$$

where $a_d = 0.7$ is an attachment dampening parameter. Hence, a machine m is increasingly likely to be sampled again each time it is sampled. That likelihood is reduced by a_d, the attachment dampening parameter, and the parameter was introduced because we noted that the job sets created with $a_d = 1$ had several machines which did not process any operations, contrary to what we observe in $\mathcal{J}_{Ind}$ where every machine processes some operations. Setting $a_d < 1$ increases the chances that every machine will process some operation, as the 1 in the

numerator remains undamped, and after some trial and error, we settled on $a_d = 0.7$ for our definition, as every machine processed some operations. If $a_d = 0$, this is the uniform routing function.

3.3.4. Preferential Flow Routing

This routing is similar to preferential attachment, but in this case, preferences develop between pairs of machines instead of for particular machines. As this distribution develops, certain machines will become more and more likely to follow certain other machines, so a degree of staging appears in job sets created this way. For instance, certain machines regularly start jobs, while others regularly complete jobs.

For our industry partner, certain processes are commonly done at the beginning of a job, while others are done commonly at the end. With this routing function, we model this property, called staging, in the random job sets. The routing functions above do not result in any staging; for uniform, triangle, and preferential attachment, machines are as likely to complete a job as they are to start a job. The preferential flow routing function models staging by creating flow relationships between pairs of machines; if some job "flows" from machine ℓ to m, then it becomes more likely that m will directly follow ℓ for other jobs. We call these job sets preferential flow job sets.

The probabilities of sampling a machine m depends primarily on the last machine added to the current job. Let j be the current job in the while loop of Algorithm 1, o_{ij} be the previous operation added to j, and m_{ij} be the machine on which o_{ij} will be processed. We next determine the probabilities for sampling a machine to assign to the operation o_{ij}. For each machine m, find and count all occurrences currently in $\mathcal{J}$ of the machines m_{ij} and m processing operations consecutively, m_{ij} then m, and denote the count $C_{m_{ij} \to m}$. Then

$$P(m) = \frac{1 + f_d C_{m_{ij} \to m}}{\sum_{\ell \in \mathcal{M}} 1 + f_d C_{m_{ij} \to \ell}}$$

where $f_d = 0.7$ is a flow dampening parameter. The value of the flow dampening parameter was arrived at similarly to the attachment dampening parameter defined in Section 3.3.3; with $f_d = 1$, several machines were not assigned any operations in the created job sets, and after some trial and error, we settled on $f_d = 0.7$.

Interestingly, the histogram of machine counts for this routing distribution, the right panel of Figure 6, is somewhat different from the histogram for the industry partner job set, the left panel of the figure. In particular, there are no machines which process between 0 and 200 operations, so the tallest bar is not close to zero, as it is for $\mathcal{J}_{Ind}$. Hence, this routing distribution fails to be more similar to the industry partner job set than the preferential attachment, though it is still of interest.

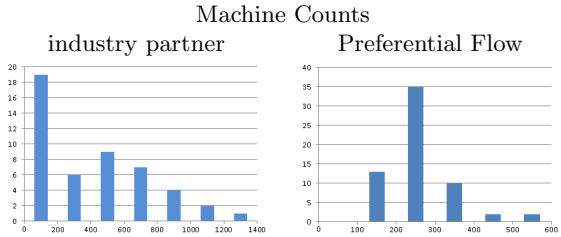

Figure 6: The left panel shows a histogram counting the number machines from $\mathcal{J}_{Ind}$ which process a number of operations processed between the bounds of each bar. The right panels show the same counts for preferential flow job sets.

3.3.5. Empirical Routing

This routing function directly models the routing one observes in $\mathcal{J}_{Ind}$. To do so, there are many dimensions to consider beyond simply the frequency with which each machine appears in $\mathcal{J}_{Ind}$; for instance, the frequency a particular machine processes the first step of a job in $\mathcal{J}_{Ind}$ is quite different from the frequency that the same machine processes the tenth step of a job. The empirical routing function is defined with this consideration in mind, and the process of sampling is described in Algorithm 2. Job sets created using this routing function are said to be empirically routed, and called empirical job sets.

In Algorithm 1, several quantities, such as job length or operation processing time, are determined using a distribution derived empirically from our industry partner's job set, $\mathcal{J}_{Ind}$, so every job set we consider here is to some degree an empirical job set. Our use of 'empirical' for the routing function described in this section, and its associated job sets, indicates that the routing function itself has been empirically derived, where the other four routing functions are not.

Algorithm 2 Sampling using the Empirical Routing Function

Require: The machine set $\mathcal{M}$.
Require: The current job j.
Require: The current operation index i.
Require: The industry partner job set $\mathcal{J}_{Ind}$.

1: **procedure** SAMPLE DISTRIBUTION ON $\mathcal{M}$
2: Sample o from the ith operations of $|O_j|$ length jobs in $\mathcal{J}_{Ind}$.
3: **if** $m_o =$ SUB **then** ▷ m_o processes o
4: Return to step 2
5: **return** m_o.

3.4. Job-Shop Scheduling as a Mathematical Program

We now formally define the job-shop problem by stating it as a mathematical program similar to that on page 86 of Pinedo (2009), using the notation defined in Section 3.2. The primary decision variables for this program are the start times, s_{ij}, of all operations, o_{ij}. A *schedule* is an assignment of a non-negative real number to all start times. A schedule is feasible if the following conditions are met by all start times:

1. operations of each job j are completed in order consecutively, and there is transportation time of $t =$ one day between machines:

$$s_{ij} + p_{ij} + t \leq s_{i+1j}$$

2. processing of job j is started after its release date:

$$r_j \leq s_{1j}.$$

3. a machine can only process one operation at a time: if the ith operation of job j and the gth operation of job h are processed on the same machine, that is $m_{ij} = m_{gh}$, then

$$s_{ij} + p_{ij} \leq s_{gh} \quad \text{or} \quad s_{gh} + p_{gh} \leq s_{ij}.$$

Our aim is to find a feasible schedule which minimizes the number of late jobs which we achieve with the following variables, constraints and objective function. A job j is not *late* if the last operation finishes on or before the due date:

$$s_{\ell j} + p_{\ell j} \leq d_j.$$

For each job, we introduce the following constraint

$$s_{\ell j} + p_{\ell j} \leq d_j + u_j \cdot L,$$

where L is a constant representing the maximum amount of time a job is allowed to be late and u_j is a binary decision variable. The u_j variables are the secondary decision variables. By taking the objective function to be $\min \sum_j u_j$, each decision variable $u_j = 1$ if and only if the job j is late, so the objective value will equal the number of late jobs. The mathematical program is summarized

below.

$$\min \sum_j u_j$$

s.t.

$$\begin{array}{rll}
u_j & \in \{0,1\} & \forall j \\
r_j & \leq s_{1j} & \forall j \\
s_{kj} + p_{kj} & \leq d_j + u_j \cdot L & \forall\, j \\
 & & \\
s_{ij} & \geq 0 & \forall\, j \text{ and } o_{ij} \in O_j \\
s_{ij} + p_{ij} & \leq s_{i+1j} & \forall\, j \text{ and } o_{ij} \in O_j \\
 & & \\
s_{ij} + p_{ij} & \leq s_{gh} & \\
 & \text{or} & \forall\, o_{ij} \text{ and } o_{gh} \text{ st } m_{ij} = m_{gh} \neq \text{SUB} \\
s_{gh} + p_{gh} & \leq s_{ij} &
\end{array}$$

3.5. Scheduling Methods

In this section, we describe the fifty five dispatching rules which we simulate, and unless cited, the proposed dispatch rule has not, to the best of our knowledge, appeared in the scheduling literature before. Each one is human implementable as each one is implementable using a printed list. These rules are triggered at a machine whenever that machine finishes processing an operation. Ties are always broken arbitrarily. The machine where the rule in question is triggered is denoted m and its queue as q_m.

We introduce some additional notation for our dispatching rule descriptions. The current time is denoted t_c. For each job j, O_j^{rem} is the sublist of O_j which lists the remaining unprocessed operations. The first operation in the list O_j^{rem} is called the *current operation* of j, and it is denoted as o_j^c while its processing time and machine are denoted p_j^c and m_j^c. If $j \in q_m$, then $m_j^c = m$. For each job, the total remaining work of j, w_j^{rem}, is the total processing of operations in O_j^{rem} plus any travel time required for those operations.

3.5.1. Earliest Due Date (EDD)

EDD selects an enqueued job at m with the earliest due date, d_j. The aim of this rule is to work on a job that is almost due, or a job that is late already (Panwalkar and Iskander, 1977).

3.5.2. Earliest Operation Due Date (EODD)

The due date for each operation of a job is calculated when that job is released. The operation due date of a job is the due date of that job's current operation, o_j^c, so the operation due date of a job will change as operations are processed.

This rule proceeds like EDD, so the job with the earliest operation due date is selected for processing.

The due dates of each operation are calculated recursively from d_j in the following way. Let h_d be the number of hours in a work day. The due date of the final operation is d_j, the due date for j. If d_{i+1j} is the due date for operation o_{i+1j}, then the due date for operation o_{ij} is $d_{ij} = d_{i+1j} - \lceil p_{ij}/h_d \rceil$. Hence, the due dates of consecutive operations are at least one day apart, allowing for the one day of travel time between machines (Panwalkar and Iskander, 1977).

3.5.3. Earliest Release Date (ERD)

Similar to EDD, ERD selects the enqueued job at m with the earliest release date, r_j. This heuristic is the first in, first out principle applied in a job-shop setting. (Pinedo, 2009).

3.5.4. Earliest Fraction Completed Date (EFCD)

The fraction completed date of a job j is a dynamic date between the release date r_j and the due date d_j, and it is calculated as follows. First, calculate the processing and transit time that has been completed on j so far, w_j^{com}. Then

$$f = \frac{w_j^{com}}{w_j^{com} + w_j^{rem}}$$

is the fraction of completed processing for j. The fraction completed date (FCD) is given by

$$\text{FCD} = fd_j + (1 - f)r_j.$$

The job in q_m with the earliest FCD is processed next on m.

This rule can be viewed as an interpolation between ERD and EDD. If FCD is a date in the future, then the processing of j can be seen as ahead of schedule; whereas, if the FCD is in the past, then j is behind schedule. By choosing the earliest FCD, the job which is most behind schedule is selected.

This rule does take some calculation, so it is not obvious that it is human implementable. However, it suffices to add another column to a work in progress report showing the FCD of each job.

3.5.5. Shortest or Longest Processing Time (SPT, LPT)

The job selected from q_m by these rules has the least or the most total work remaining, that is the job with the minimum or maximum w_j^{rem} (Panwalkar and Iskander, 1977).

3.5.6. Shortest or Longest Operation Processing Time (SOPT, LOPT)

For these two rules, a job is chosen from q_m if it has the least or most processing time for its current operation, that is the minimum or maximum p_j^c for $j \in q_m$ (Panwalkar and Iskander, 1977).

3.5.7. Least or Most Remaining Operations (LRO, MRO)

A job is selected for processing if it has the least or the most operations remaining, that is the minimum or maximum $|O_j^{rem}|$ for $j \in q_m$ (Panwalkar and Iskander, 1977).

3.5.8. Shortest Next Queue (SNQ)

For each job $j \in q_m$, let o_j^{c+1} be the operation which follows o_j^c in O_j^{rem}, and let $n_j = m_j^{c+1}$ be the machine where that o_j^{c+1} is processed. Then we find the total amount of processing of all the current operations of jobs currently enqueued at the next machine n_j:

$$T_j = \sum_{k \in q_{n_j}} p_k^c.$$

The job $j \in q_m$ with the minimum T_j is selected for processing on m. By selecting the shortest next queue, this heuristic tends to avoid sending jobs on to "overloaded" machines, so it tends to balance the work loads of all machines (Panwalkar and Iskander, 1977).

3.5.9. First in, First or Last out (FIFO, FILO)

For this rule, we keep track of the arrival order of jobs into every queue, and the job which arrived first, or which arrived most recently, is selected for processing (Panwalkar and Iskander, 1977).

3.5.10. Maximum Work per Day (MWpD)

The work per day for a job j is calculated as the ratio of the remaining processing w_j^{rem} over the number of days remaining before d_j if the due date is not passed. If d_j is passed, then the work per day equals w_j^{rem}. Hence, the WpD is calculated as

$$\text{WpD} = \frac{w_j^{rem}}{\max(1, \lceil d_j - t_c \rceil)}.$$

The job in q_m with the maximum WpD is selected for processing.

3.5.11. Least Slack (LS)

The slack a job has is the amount of time before its due date less the amount of remaining work w_j^{rem}:

$$\text{slack}(j) := (d_j - t_c) - w_j^{rem}.$$

The slack of a job is positive only if j can be completed before its due date (Panwalkar and Iskander, 1977).

3.5.12. Least Slack over Remaining Operations (LS/RO)

The quantity S/RO is calculated as

$$\text{S/RO} = \frac{(d_j - t_c) - w_j^{rem}}{|O_j^{rem}|} = \frac{\text{slack}(j)}{|O_j^{rem}|}.$$

The job in q_m with the least S/RO is selected. If two jobs are late and have the same slack, then the one with fewer operations is selected; whereas, for two early jobs, this rule selects the job with the most remaining operations (Panwalkar and Iskander, 1977).

3.5.13. Least Slack over Operation Processing Time (LS/OPT)

This is similar to LS/RO where the slack is calculated for each job, and then a ratio is formed. Here, the denominator of the ratio is the processing time of the current operation:

$$\text{S/OPT} = \frac{(d_j - t_c) - w_j^{rem}}{p_j^c} = \frac{\text{slack}(j)}{p_j^c}.$$

The job in q_m with the least ratio is selected for processing.

The aim of this rule is to handle jobs differently depending on whether the slack is positive or negative and the size of the current operation. Firstly, any job with negative slack will be processed before a job with positive slack. Secondly, if slack is positive, larger current operation lengths decrease the ratio compared to smaller operation lengths, whereas if slack is negative, larger operations increase the ratio over smaller operations. Hence, for jobs with positive slack, jobs with longer current operations are processed first, while for jobs with negative slack, jobs with shorter current operations are processed first.

*3.5.14. Least Slack times Operation Processing Time (LS*OPT)*

For each job, calculate the product

$$\text{S*OPT} = \text{slack}(j) * p_j^c.$$

The job in q_m with the least product is selected for processing.

The aim is very similar to LS/OPT. The jobs with negative slack will be processed before jobs with positive slack, but the the current operation length has the opposite affect on selection because the quantity calculated is a product instead of a ratio.

3.5.15. Least Slack plus a late rule (LS lr)

This rule is formed as a combination of two rules. The bounded least slack rule is used to find all the jobs in q_m with the least bounded slack, where we bound slack, defined for rule 3.5.11, by zero:

$$\max(0, \text{slack}(j)).$$

The second rule, the late rule, is used to select from among those jobs. Due to the bound at zero, the primary effect of this rule is that late jobs with 0 slack are selected using a different rule than jobs with positive slack, though it also breaks ties between jobs with the same positive slack. If the late rule is the least slack rule, then the combination is equivalent to the least slack rule.

3.5.16. Most Tardy plus an early rule (MT er)

This rule uses the tardiness of job j, namely the amount of time a job is past its due date:

$$\max(0, t_c - d_j).$$

This rule operates similarly to LS lr above; first, all jobs in q_m with the highest tardiness are found, and then a second rule is used to select a job for processing from among those found jobs. This rule also has a similar primary effect as LS lr in that tardy and non-tardy jobs in q_m are selected using different rules and the tardy jobs are always selected over the non-tardy jobs.

3.5.17. Most Tardy over Remaining Processing Time (MT/PT)

Calculate the tardiness of each job as $t_c - d_j$, allowing negative values. Divide this value by the total processing time remaining for that job,

$$\frac{t_c - d_j}{w_j^{rem}},$$

and select the job with the highest such ratio for processing. The ratio calculated for this dispatching rule is also known as the critical ratio. (Pinedo, 2009)

3.5.18. Most Tardy over Operation Processing Time (MT/OPT)

This is similar to MT/PT, but the ratio is formed using the processing time of the current operation:

$$\frac{t_c - d_j}{p_j^c}.$$

(Panwalkar and Iskander, 1977)

3.5.19. Multiple Rules

Lastly, we consider assigning multiple rules to the job-shop machines. To do so, we use the departments, groups of machines, we observed in our industry partner's factory. There are five departments, and they have 27%, 26%, 19%, 15%, and 13% of the total number of machines respectively. For a scheduling method with multiple rules, each of the five departments is assigned a dispatching rule, and at least two different rules are assigned.

Now, all the job sets we create have the same number of machines, and there is a canonical mapping of those machines to the machine in the industry partner job set $\mathcal{J}_{Ind}$. With that mapping, we are able to use the same departments on all the job sets by mapping the machines of $\mathcal{J}_{Ind}$ to the machines in all the other job sets $\mathcal{J}$.

However, the original departments our industry partner created were not created arbitrarily; they were created using explicit and implicit forms of analysis. By using the above mapping, the department assignments made for the random job sets are essentially random expect for the size of these departments. To correct this problem, an analysis similar to that under taken by our industry partner would be required to similarly assign the machine in the random job sets to departments. This, unfortunately, goes beyond that scope of this thesis.

4. industry partner Results

We now explore the results of the deterministic simulations run on our industry partner's production data, the job set $\mathcal{J}_{Ind}$. In the Section 4.1, we show that the human implementable rule our industry partner currently uses, EODD, is the best scheduling method for them to use. The best multiple rule method in Section 4.2 does not out perform EODD enough to warrant its use, as determined by our industry partner. There are also several other candidate scheduling methods which significantly out perform EODD on the percentage of late jobs KPIs, but the trade-off of large increases to the maximum lateness is not acceptable to our industry partner. (Industry-Partner, 2013)

4.1. Single Rule Performance

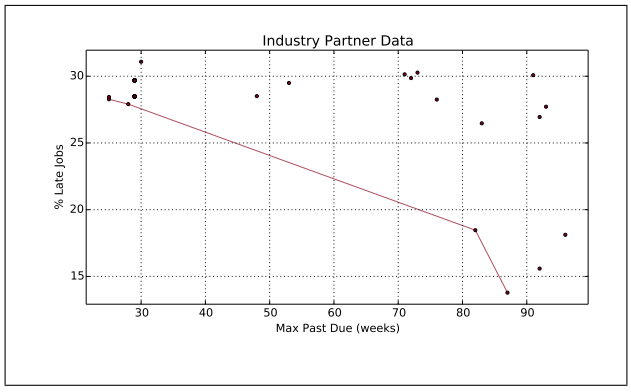

Figure 7: Simulation results for single dispatching rules plotted by their performance on our KPIs: maximum lateness versus percentage of late jobs.

The simulation results shown in Figure 7 show the performance of all fifty five single dispatching rules on $\mathcal{J}_{Ind}$. In each simulation, one dispatching rule is used on all machines, and that rule has a point in Figure 7 with coordinates given by how the rule performed on our two KPIs: the x-coordinate is the maximum lateness, in weeks, across all completed jobs, and the y-coordinate is the percentage of simulated jobs completed late. There is no single best performing dispatching rule; instead, there is a trade-off of performance for the two KPIs, so the best performing rules form a KPI frontier, which is indicated by the line in the figure. The four rules on the KPI frontier are listed in Table 1 with their coordinates. The results in the figure and table are for the performance of a single simulation on $\mathcal{J}_{Ind}$ using one rule, as the simulations are deterministic.

The earliest operation due date rule, the rule currently employed by our industry partner, is on the KPI frontier, so it is one of the best performing rules on the job set $\mathcal{J}_{Ind}$. Hence, EODD is a fine choice for our industry partner's job-shop.

Rule	Max Late	% Late
EODD	25.0 weeks	28.27%
EDD	28.0 weeks	27.90%
SNQ	82.0 weeks	18.46%
SOPT	87.0 weeks	13.78%

Table 1: Best performing single rules on the industry partner data.

Our simulations also indicate that performance trade-offs for the other rules on the KPI frontier are not acceptable for our industry partner. The rule EDD, the second rule listed in Table 1, lengthens the maximum lateness by three weeks over the score for EODD, a 12% increase; the trade-off is a reduction of the percentage of late jobs by 0.4%, a relative decrease of only 1.4%. This is not a favourable trade-off for our industry partner (Industry-Partner, 2013).

Further, the rules SNQ or SOPT greatly reduce the number of late jobs compared to EODD; Table 1 shows that the percentage of late jobs is 10% and 15% less, a relative reduction of 35% and 51%, respectively. However, to use SNQ or SOPT, our industry partner would have to accept a maximum lateness over three times as large, over 80 weeks past due in each case. In real terms, jobs completed over 80 weeks past due are essentially abandoned, and our industry partner does not wish to refuse work or subcontract out entire jobs. Hence, SNQ and SOPT are not acceptable dispatching rules for their job-shop. (Industry-Partner, 2013)

4.2. Multiple Rule Methods Results

For these results, the simulations assign one of eight dispatching rules to each of five departments; Table 2 lists the eight rules, and the five departments are described in Section 3.5.19. The eight rules were selected because they are the eight most commonly appearing rules on the KPI frontiers discussed in Sections 4.1 and 5.1, the single dispatching rule KPI frontiers. The remaining dispatching rules from Section 3.5 were not assigned in any of the multiple rule simulations as time did not permit; the number of possible assignments to five departments grows factorially with additional rules, and each simulation takes an average of 1 minute.

EODD	SOPT	EDD	MT EFCD	SNQ	EFCD	FIFO	MRO

Table 2: The eight rules assigned to the five departments from Section 3.5.19.

The performance of all methods, single and multiple together, are shown in Figure 8. As in Section 4.1, there is a KPI frontier in the figure indicating again that there is a trade-off between the two KPIs. In Table 3, the nine points on the KPI frontier are listed.

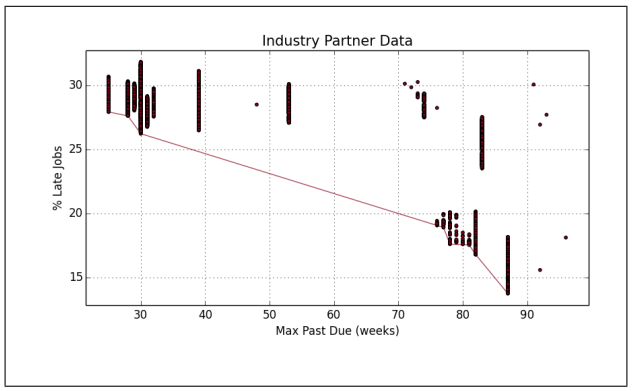

Figure 8: Results for simulations using multiple dispatching rules.

Dept 1	Dept 2	Dept 3	Dept 4	Dept 5	Max Late	% Late
EODD	EODD	EODD	EODD	SOPT	25.0	27.93
EODD	EDD	EODD	EODD	SOPT	28.0	27.63
SOPT	EDD	EODD	EODD	SOPT	30.0	26.23
MT EFCD	SNQ	EDD	MT EFCD	SNQ	76.0	19.06
MT EFCD	SNQ	SNQ	SOPT	EDD	77.0	18.91
EDD	SNQ	SNQ	EFCD	SNQ	78.0	17.60
SOPT	SNQ	SOPT	EODD	SNQ	81.0	17.54
EODD	SNQ	EODD	SOPT	EODD	82.0	16.79
SOPT	SOPT	EODD	SOPT	SOPT	87.0	13.74

Table 3: The Parteo frontier of all industry partner data simulations. The rules are assigned to the departments defined in Section 3.5.19.

Comparing the numbers in Table 3 to Table 1 in Section 4.1, it is clear that, where multiple rule methods out performed a single dispatching rule, the degree they out performance is very small. For instance, the first entry of Table 3 has a very similar performance to EODD: 25 weeks maximum lateness for both and 27.93% late jobs versus 28.27% late jobs. The difference in the percentage of late jobs is 0.34%, only a 1% relative difference. In Figure 9, the proximity of the two frontiers is evident. With a similar analysis as in Section 4.1, the first entry of Table 3 is a good choice for our industry partner. However, the decrease in the percentage of late jobs is not enough, at 0.34%, to justify the extra effort of maintaining two different dispatching rules in two different departments. (Industry-Partner, 2013) EODD by itself remains the better choice overall.

There is a question of whether more rules used in combination improves job-shop performance. Figure 10 shows KPI frontiers for all simulations separated by the number of rules the scheduling method employs; for instance, the 2 rules frontier is the KPI frontier for every multiple rule method which assigned exactly two rules to the five departments. The two, three and four rule frontiers are nearly indistinguishable across their entire length. The five rule KPI frontier is

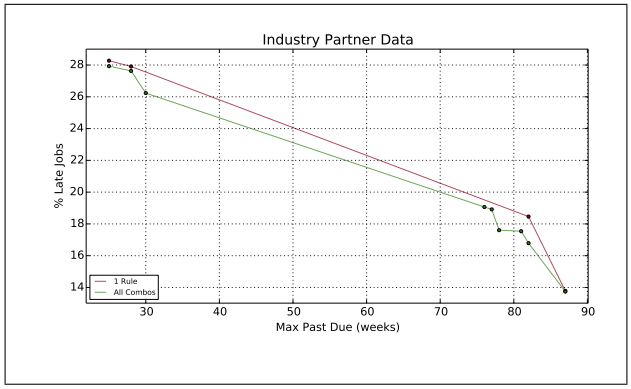

Figure 9: The KPI frontiers for the single rule methods and for all the methods run on the industry partner data simulations.

above each of those three frontiers, as no five rule method is on the KPI frontier. Overall, there is very little for our industry partner to gain from using multiple rule methods over the best single rule methods, and what little can be gained is almost entirely achieved using two rule methods.

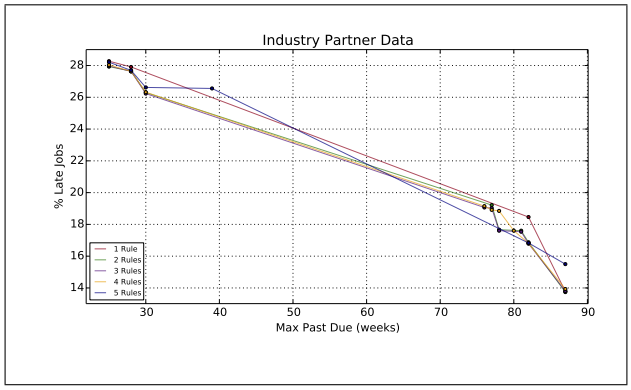

Figure 10: The KPI frontiers for scheduling methods using 1 to five different rules.

5. Generalized Results

In this section, we show results applicable to other job-shops. While our simulations are deterministic, the creation of the job sets is a random process, so for statistical significance, fifty different job sets were simulated for each of the five routing distributions described in Section 3.3: empirical routing, preferential attachment, preferential flow, triangle routing, and uniform routing. The performance of the rules are assessed as averages of the performances on all fifty job sets for given routing distribution. We present the results for each routing distribution using charts similar to those in Section 4, with the addition of 95% confidence intervals for the calculated averages. We find that EODD is consistently one of the best performing rules regardless of routing distribution. We also find that the multiple rule combinations out perform EODD, but on average, the combinations do not out perform single rules enough to warrant the complexity of their implementation for any of the routing distributions we consider.

5.1. Single Rule Results

In Figure 11, we see a KPI frontier for each routing distribution. The points along each frontier indicate the performance of one of the best performing single rules for the corresponding routing distribution. The frontier for the industry partner data in Figure 11 is the same frontier as that shown above in Figure 7. In this section, we are showing results for scheduling methods which use a single rule across all departments, so the department structure can be ignored.

Confidence intervals are shown as rectangles about each point, and they indicate the 95% confidence interval for the averages represented by the x and y coordinates of the point. We refer to these rectangles as *confidence rectangles* below.

The figure also shows the extent to which dispatching rule performance is affected by the routing distribution. Recall from Section 3.2, the only parameter which changes for each routing distribution is the routing function used to create the job sets, so the only difference is the ways jobs flow through the job-shop. The other properties of the job sets, such as job length and due date, are all sampled from the same empirical distributions derived from $\mathcal{J}_{Ind}$. Despite this consistency across all job sets, Figure 11 shows quite a large variance of both the position and length of the KPI frontiers. In the following five subsections, we detail the impact routing distribution has on the fifty five single dispatching rules we simulated, as well as show that EODD is the best performing rule for most routing distributions.

5.1.1. Empirical Routing

From the definition, it appears that the empirical routing distribution, of all the routing distributions considered here, should be the best model of the industry

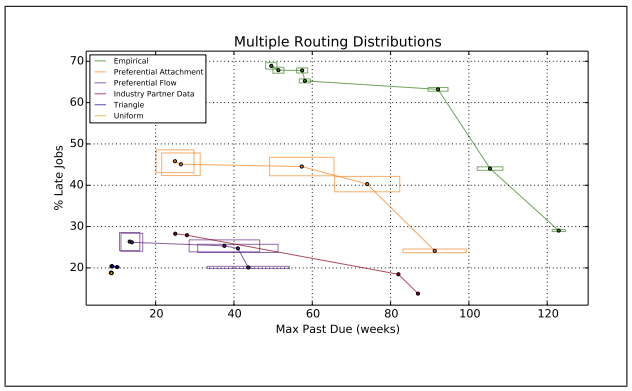

Figure 11: The single rule KPI frontiers for each routing distribution.

partner data, as it is the only routing distribution which uses $\mathcal{J}_{Ind}$ to determine machine order. Indeed, we expected the performance of dispatching rules on empirically routed job sets to be similar to their performance on $\mathcal{J}_{Ind}$.

However, the large difference in position and length of the Empirical and industry partner data frontiers in Figure 11 indicated that the dispatching rules show severely degraded performance on empirically routed job sets. The degraded performance occurs despite the facts that the number of machines is constant for all the simulated job sets and the number of operations in every job set is within 0.4% of the number of operations in $Op(\mathcal{J}_{Ind})$.

While we are uncertain exactly why the performance degraded, we suspect it is due to the decoupling of one or more correlated features of $\mathcal{J}_{Ind}$. For instance, job length could be seasonally correlated if our industry partner has seasonal clients. There is also the possibility that our industry partner uses various soft strategies, which were not described to us during our interviews and job shadowing, to coordinate release dates, schedule subcontracting, account for expected demand or other features of production. If so, the job set $\mathcal{J}_{Ind}$ has more "helpful" structure than we assume, and all the random job sets we create do not include that structure. To maintain confidentiality of the data used, we do not analyze these possibilities in detail.

Note that the confidence intervals are less than 3% either side of all the averages in table 4, so the confidence rectangles in Figure 12 are small in area. These features indicate that the simulated performance of each of the 55 single dispatching rules is consistently bad across all fifty empirically routed job sets. In fact, many of the rules appear to have the exact same performance on every empirically routed job set; there are 18 rules listed in Table 4 that have the same average performance as ERD. This is indicated by the darker rectangle about the point $(58.12, 65.26)$ in Figure 12. ERD is included on the KPI frontier over the others because it is the simplest of the rules with that performance. The confidence rectangle for the Least Slack (LS) dispatching rule overlaps with the

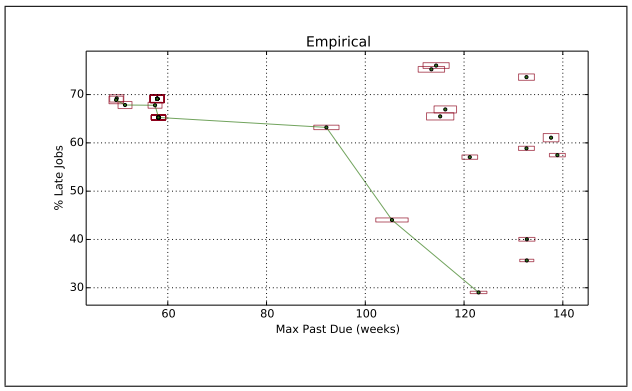

Figure 12: Single rule performances on the Empirical routing distribution.

rectangle for EODD, and it is the only other dispatching rule to overlap the rectangles on the frontier.

Rule	Max Late	% Late
EODD	49.54 weeks ±1.45	68.88% ±0.78
EDD	51.34 weeks ±1.41	67.84% ±0.73
EFCD	57.42 weeks ±1.41	67.79% ±0.62
ERD (and 18 others)	58.12 weeks ±1.43	65.26% ±0.48
FIFO	92.12 weeks ±2.56	63.20% ±0.46
SNQ	105.40 weeks ±3.28	44.04% ±0.42
SOPT	122.94 weeks ±1.69	29.03% ±0.24

Table 4: Best performing single rules for the empirical routing distribution.

Rule	Max Late	% Late
LS	49.66 weeks ±1.43	69.21% ±0.76

Table 5: The only rule which partially overlaps confidence rectangles on the empirical routing distribution KPI frontier.

Despite the worsening of overall performance compared to the industry partner job set, the relative performances of the best rules is very similar. The four rules on the KPI frontier for the industry job set, Table 1, are all on the KPI frontier for the empirical job sets listed in Table 4 in the same order from left to right on their respective frontiers. Hence, there is a similar trade-off between the KPIs along the frontiers: EODD minimizes the number of weeks all jobs are past due, SOPT minimizes the percentage of late jobs, and the rules in between on the frontier are various performance mixtures. We can quantify the trade-off using the concept of elasticity from economics.

Table 6 shows the 'percentage of late jobs' (PLJ) elasticity of the 'maximum lateness' (ML) for the KPI frontier for empirically routed job sets. Here, the

EDD	-2.34
EFCD	-9.18
ERD	-2.95
FIFO	-6.99
SNQ	-1.64
SOPT	-1.05

Table 6: The PLJ arc-elasticity of ML with EODD for the empirical routing distribution.

elasticities are calculated as arc-elasticities for a pair of KPI frontier points, $(\mathrm{ML}_1, \mathrm{PLJ}_1)$ and $(\mathrm{ML}_2, \mathrm{PLJ}_2)$, by the formula

$$\left(\frac{\mathrm{ML}_2 - \mathrm{ML}_1}{\mathrm{PLJ}_2 - \mathrm{PLJ}_1}\right) \cdot \left(\frac{(\mathrm{ML}_2 + \mathrm{ML}_1)/2}{(\mathrm{PLJ}_2 + \mathrm{PLJ}_1)/2}\right)^{-1}.$$

In economics, price elasticity of demand is generally calculated at a point, $pt = (P, Q)$, on a continuous demand curve as the slope at pt divided by the ratio of the coordinates:

$$\left(\frac{dP}{dQ}\right) \cdot \left(\frac{P}{Q}\right)^{-1}.$$

In the case of point-wise data, arc-elasticities are calculated instead; the slope is calculated for the arc between a pair of points, and the midpoint of that arc is used for the ratio of coordinates.

The purpose of elasticity is to quantify the percentage change of one quantity, the ML KPI in our case, for each 1% decrease of another quantity, the PLJ KPI. Hence, an arc-elasticity for two points on a KPI frontier indicates the exact trade-off a job-shop makes in choosing one of the scheduling methods over the other. For example, the performances of EODD and EDD have an arc-elasticity of -2.34; this means that if a job-shop switched to EODD after having used EDD and the job-shop experienced an $\ell\%$ relative increase in the percentage of late jobs after the switch, then they can expect a simultaneous, relative decrease of $\ell * 2.34\%$ in the maximum lateness KPI.

We assume that the two KPIs are of equal importance and our goal is to minimize both simultaneously. An arc-elasticity of less than -1 for a pair of KPI frontier points indicates that the rule further to the left in Figure 12 is the better choice for minimizing both KPIs simultaneously. The pairwise elasticities with EODD in Table 6 are all less than -1. Hence, we recommend EODD as the best performing single dispatching rule on the empirically routed job sets, with the caveat that LS possibly out performs EODD on some job sets.

5.1.2. Preferential Attachment

In Figure 11, we see that the KPI frontier for the preferential attachment (PA) job sets is between the frontiers for the industry partner data and the empirical

job sets, so PA job sets are "easier" to process than the empirically routed job sets though more "difficult" than the industry partner data. It may be possible to adjust the attachment dampening parameter a_d, defined in Section 3.3.3, when creating preferential attachment job sets and so create job sets which post dispatching rule performances much closer to those observed for the industry partner data. However, to maintain manageable project scope, we forewent a comparison of different a_d values, including such a comparison in our list of future work to consider in Section 7.3.4.

Note the higher degree of variance in the averages indicated by the large confidence rectangles about the points on the PA frontier in Figure 11, which are greater than the confidence rectangles on the empirical KPI frontier. The variance increase is observed for all 55 dispatching rules, as all the rectangles in Figure 13 are of comparable size to those on the KPI frontier. Hence, the random process used to create the preferential attachment job sets introduces more variation in the features most affecting dispatching rule performance than does the process used to create the empirically routed job sets.

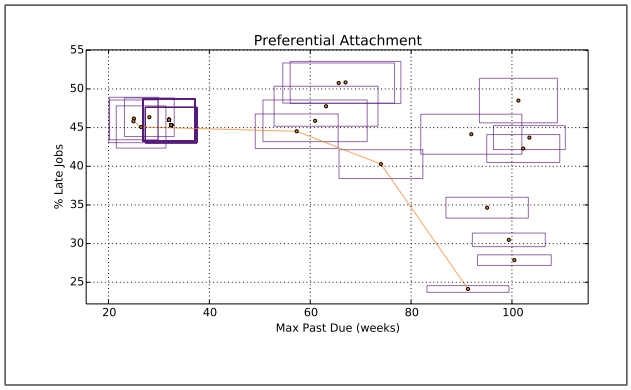

Figure 13: Single rule performances for the preferential attachment routing distribution.

Despite these differences, the list of rules on the KPI frontier, Table 7, is rather similar to the frontiers for the empirically routed job sets and the industry partner data: EODD, EDD and SOPT appear on all three in the same order, and FIFO appears on the empirical frontier. Hence, there is a similar trade-off between our KPIs: choosing EODD minimizes the maximum lateness KPIs, SOPT minimizes the percentage late, and the other rules are mixtures. Table 7 also shows that choosing a rule which increases one or the other KPI also increases the amount of performance variance of that KPI. This is visible in Figure 13 as the change from squares to thin rectangles when sweeping over the figure from left to right.

The arc-elasticity for our two KPIs are shown for EODD in Table 8. As we saw for the empirically routed job sets, all the elasticities for EODD are less than -1, so EODD is again the best choice to minimize both KPIs simultaneously. There

Rule	Max Late	% Late
EODD	24.94 weeks ±4.84	45.81% ±2.76
EDD	26.46 weeks ±4.92	45.08% ±2.73
FIFO	57.32 weeks ±8.22	44.53% ±2.24
MRO	74.02 weeks ±8.32	40.28% ±1.86
SOPT	91.28 weeks ±8.13	24.13% ±0.44

Table 7: Best performing single rules for the preferential attachment routing distribution.

are also two dispatching rules, LS and EFCD, with performances which overlap EODD, so EODD might be out performed by one or both of these dispatching rules on some job sets. Table 9 shows a list of the rules with performances overlapping points of the preferential attachment job sets KPI frontier.

EDD	-3.66
FIFO	-27.66
MRO	-7.71
SOPT	-1.84

Table 8: The PLJ arc-elasticity of ML with EODD for the preferential attachment routing distribution.

Rule	Max Late	% Late
LS	25.04 weeks ±4.83	46.17% ±2.75
EFCD	28.10 weeks ±4.93	46.35% ±2.49
LS EDD (and 16 others)	32.00 weeks ±5.18	46.00% ±2.73
ERD (and 18 others)	32.42 weeks ±5.10	45.33% ±2.28
LS/RO	60.96 weeks ±10.33	45.88% ±2.69

Table 9: The rules with performances overlapping the KPI frontier for preferential attachment job sets.

5.1.3. Preferential Flow

The features of the average performances of the fifty five dispatching rules on the preferential flow (PF) job sets are very similar to the features mentioned in Section 5.1.2 about the preferential attachment job sets. The preferential flow frontier in Figure 11 shows improved performance over the industry partner data, preferential attachment, and empirical KPI frontiers. The confidence intervals in Figure 14 are as large as those for preferential attachment job sets shown in Figure 13, so the random process for creating the preferential flow job sets introduces similar variability in job set features which affect dispatching rule performance. The confidence rectangles also show the trade-off between our KPIs; we go from tall rectangles to squat, long rectangles when sweeping from left to right in Figure 14.

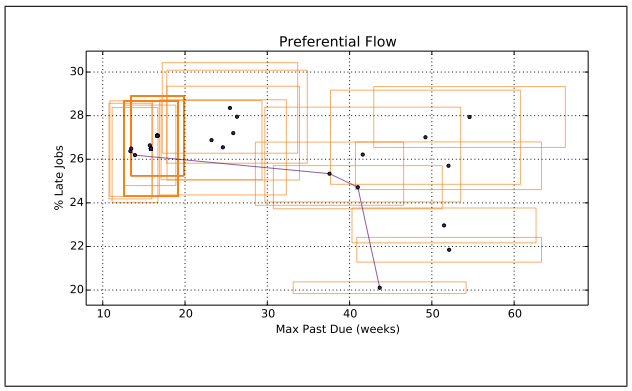

Figure 14: Single rule performances on the preferential flow routing distribution.

Rule	Max Late	% Late
EODD	13.36 weeks ±2.60	26.37% ±2.19
EDD	13.92 weeks ±2.78	26.19% ±2.18
SNQ	37.56 weeks ±9.01	25.34% ±1.45
MRO	41.00 weeks ±10.27	24.72% ±0.99
SOPT	43.66 weeks ±10.51	20.11% ±0.27

Table 10: Best performing single rules for the preferential flow routing distribution.

The rules listed on the KPI frontiers in Tables 7 and 10 are nearly all the same, with the only change being FIFO is exchanged with SNQ. Expressing the trade-off along the KPI frontier using elasticities, shows that EODD is the best rule for minimizing both KPIs simultaneously, as all the elasticities in Table 11 are less than -1. The dispatching rules LS and EFCD both have confidence rectangles which overlap the rectangle for EODD, so there is a chance they could out perform EODD on some job sets.

EDD	-5.85
SNQ	-23.74
MRO	-15.70
SOPT	-3.94

Table 11: The PLJ arc-elasticity of ML with EODD for the preferential flow routing distribution.

The biggest difference between the results for preferential flow and the other job sets discussed above is the degree of overlapping we see in Figure 14; every rectangle, save the one around SOPT, overlaps another rectangle. The length of Table 12 shows more evidence of the higher amount of overlap, as there are many more rules listed. There is an overall narrowing of the range of possible performances in Figure 14 which, when coupled with high variance, results in

Rule	Max Late	% Late
LS	13.46 weeks ±2.60	26.49% ±2.19
EFCD	15.72 weeks ±3.16	26.64% ±1.84
LS EDD (and 17 others)	15.86 weeks ±3.29	26.49% ±2.19
ERD (and 18 others)	16.64 weeks ±3.22	27.08% ±1.83
FIFO	23.22 weeks ±6.13	26.88% ±1.82
LS/RO	24.60 weeks ±7.72	26.55% ±2.18
MT/PT	25.86 weeks ±8.05	27.02% ±2.15
LS*OPT	41.86 weeks ±11.90	26.22% ±2.17
LRO	51.48 weeks ±11.19	22.97% ±0.80
LPT	52.02 weeks ±11.31	25.70% ±1.09
SPT	52.10 weeks ±11.23	21.85% ±0.57

Table 12: The rules with performances overlapping the KPI frontier for preferential flow job sets.

the observed overlapping and the shorter KPI frontier.

5.1.4. Triangle Routing

The KPI frontier for the triangle job sets is quite short in length in Figure 11, particularly in comparison to the empirical frontier. Indeed, the range of dispatching rule performances is also quite restricted, with all confidence rectangles in a 6% by 9 week area in Figure 15. None of the KPI frontiers discussed above and shown in Figure 11 could fit into an area of similar size. Even the worst performing rule, shown at the top right of Figure 15, posts a better performance, in both KPIs, on the triangle job sets than any rule posts on the job sets so far discussed.

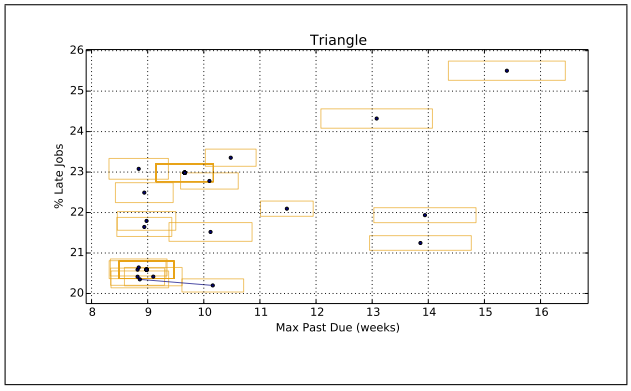

Figure 15: Single rule performances for the triangle job sets.

However, the relative performance of the best rules is similar to the job sets discussed in the previous sections. The KPI frontier, listed in Table 13, contains EODD, EDD and SOPT, as have all the previous frontiers discussed so far. The pairwise, arc-elasticities, shown in Table 14, indicate that EODD is again the

best dispatching rule for simultaneously minimizing both KPIs, and LS, along with 19 other rules, overlaps EODD's performance, indicating they may actually out perform EODD on some job sets. Essentially, the story of the previous sections was compressed into a much smaller performance range.

Rule	Max Late	% Late
EODD	8.82 weeks ±0.48	20.41% ±0.21
EDD	8.86 weeks ±0.51	20.35% ±0.21
SOPT	10.16 weeks ±0.55	20.20% ±0.16

Table 13: Best performing single rules for the triangle routing distribution.

EDD	-1.4
SOPT	-13.31

Table 14: The PLJ arc-elasticity of ML with EODD for the triangle routing distribution.

Rule	Max Late	% Late
LS	8.82 weeks ±0.51	20.59% ±0.22
LS EDD (and 17 other)	8.98 weeks ±0.49	20.59% ±0.22
LS*OPT	9.10 weeks ±0.51	20.42% ±0.23

Table 15: The rules with performances overlapping the KPI frontier for triangle job sets.

5.1.5. Uniform Routing

In Figure 11 above, the KPI frontier for the uniform routing distribution appears as a single point. The frontier actually consists of three points, but due to the scale of Figure 11 and the proximity of the points to each other, the frontier is shown as a single point. The performances of all the rules on the uniformly routed job sets fall into a very narrow range in Figure 16: between 8 and 10.5 maximum weeks late and between 18.5% and 21% late jobs. This is an even smaller range than we noted for the triangle job sets. The dispatching rules perform the best on the uniformly routed job sets of all the job sets we considered.

Table 16 also shows a few other differences; it is the only KPI frontier without SOPT on it, with LS on it, and where EODD is not the rule with the best average performance. This is also the only frontier where all the confidence rectangles for the rules on the frontier all overlap each other, and there are twenty one other rules, listed in Table 17, which overlap all three rules on the frontier. With another set of fifty uniformly routed job sets, the relative performances shown in Figure 16 and Tables 16 and 17 could be entirely changed.

In addition, the fine distinctions of performance in Figure 16 are unlikely to be realizable in an actual uniformly routed job-shop due to the complexity and the

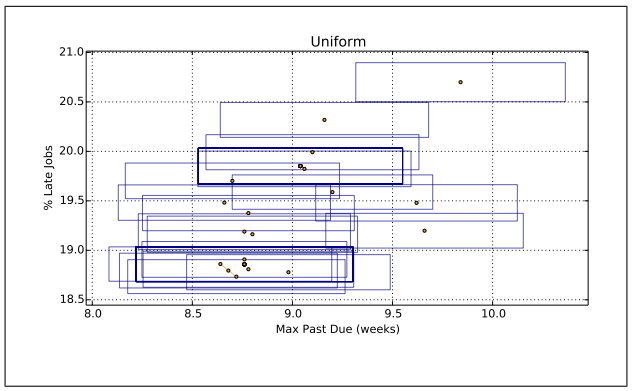

Figure 16: Single rule performances for the uniform job set.

large number of factors affecting production, so because all the rules perform so similarly, it reasonable to suggest that a real, uniformly routed job-shop could use any of the fifty five dispatching rules with equal success. Therefore, the simplicity of a rule is a much greater consideration than the performance differences we observed in our simulations, and the rule EDD is among the simplest of the rules we consider, requiring only a list of due dates for implementation.

Rule	Max Late	% Late
LS	8.64 weeks ±0.56	18.86% ±0.17
EODD	8.68 weeks ±0.54	18.80% ±0.18
EDD	8.72 weeks ±0.54	18.73% ±0.17

Table 16: Best performing single rules for the uniform routing distribution.

Rule	Max Late	% Late
LS EDD (and 17 other)	8.76 weeks ±0.54	18.86% ±0.18
LS/RO	8.76 weeks ±0.51	18.91% ±0.18
LS*OPT	8.78 weeks ±0.53	18.81% ±0.18
SOPT	8.98 weeks ±0.51	18.78% ±0.18

Table 17: The rules with performances overlapping the KPI frontier for uniform job sets.

5.2. Routing Function Multiple Rule Methods Results

In this section, we explore the performance gains achieved by using multiple rule methods. The simulations we ran are setup as follows. The sixty two machines used for all the jobs sets are gathered into five departments as discussed in Section 3.5.19. The scheduling methods we simulated are all possible assignments to five department of two or more of the the rules EODD, EDD, SOPT,

and FIFO. These four rules were selected as they appear on a majority of the KPI frontiers above. We limit the number of rules to four due to computing time constraints. The increased number of simulated scheduling methods increases the number of points on the KPI frontier, as is evident in Figure 17. The KPI frontiers for the five routing distributions are listed in Appendix C for completeness.

We assess all the scheduling methods by the same two KPIs as we assessed the results in Section 5.1 with, but in this section, we compare the multiple rule performances to the performance of the single rules on each routing distribution, specifically to EODD as it is consistently the best performing single rule across routing distributions. We also examine whether using more rules in combination offers any performance gains. We find that for all routing distributions the best multiple rule methods do out perform the best single rules, but the performance differences are not statistically significant. As such, we cannot recommend the implementation of multiple rule methods for any of the routing distributions. Further, we find no statistical performance gain for using more than two rules in combination on any of the routing distributions.

5.2.1. Comparing KPI Frontiers

For each routing distribution, we compare the single rule KPI frontier to the multiple rule frontier. Figure 17 shows all the KPI frontiers together, and they show the same progression that the single rule KPI frontiers do in Figure 11: the empirical routing distribution at the top right, and the uniform distribution showing as a single point in the bottom left. The frontiers are also similarly positioned and have a similar length. The multiple rule methods collectively perform very much like the single dispatching rules.

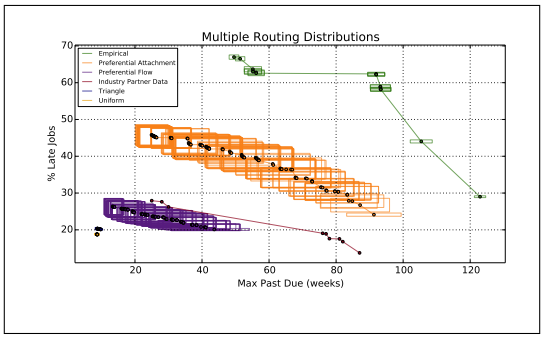

Figure 17: The KPI frontiers over all methods for each routing distribution.

Figure 18 displays the KPI frontiers for each routing distribution. The red frontier in each figure is the single dispatching rule KPI frontier for that routing distribution; each of those are discussed in detail in Section 5.1. The green frontier is the KPI frontier for all simulated scheduling methods, single rules

and multiple rules together. In all the figures, we see that the average point of the green frontier is below the red frontier, so the best performing multiple rule methods do out perform the best single rules on for each routing distribution in average terms. However as the confidence rectangles overlap for all but the empirical distribution, the same cannot be said statistically.

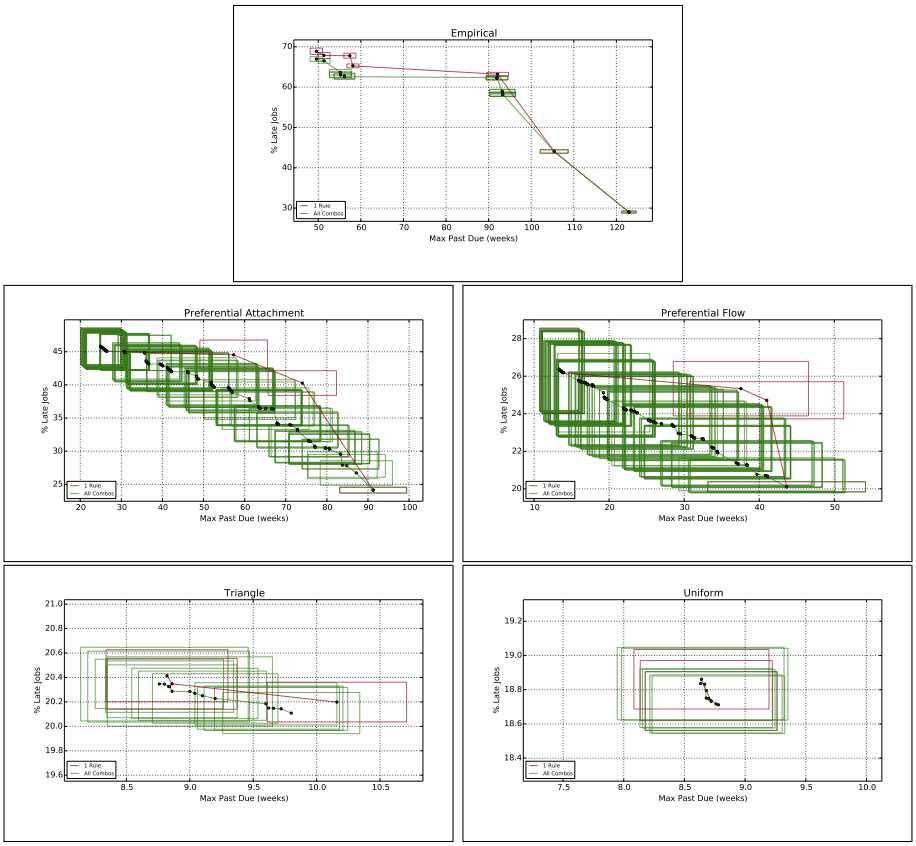

Figure 18: The KPI frontiers for single rule simulations (red) and for all simulations (green) for each routing distribution.

The empirical routing frontiers, the top row of Figure 18, are visually very similar. The main difference is that the green frontier is a few percentage points below the red frontier.

The multiple rule frontiers for the preferential attachment and flow routing distributions, the middle row of Figure 18, both have significantly more points on them than the single rule frontiers, so they are a better interpolation of the end points than the single rule frontiers. It appears the end points of the green and red frontiers in both figures are the same.

Finally, for the triangle and uniform routing distributions, the bottom row of

Figure 18, neither of the frontier pairs show large statistical difference. The confidence rectangles of the green frontiers entirely contain the red frontiers. In addition, the variation of the values is quite narrow. While the best average performances for the multiple rule methods are better than the averages of the single rule average performances, we cannot be confident that the relative positions of the two frontiers will remain as shown in the figures if the simulation sample size were increased.

5.2.2. Comparing to EODD

As was noted in Section 5.1, the dispatching rule EODD is the best rule for simultaneously minimizing both of our KPIs for all but one of the routing distributions, and for that exception, the uniform routing distribution, EODD's performance is statistically indistinguishable from the best performing rule, LS. EODD is also the rule used by our industry partner. For these reasons, we use the performance of EODD on each of the routing distributions as the bench mark for the performances of the multiple rule methods.

The multiple rule method we compare EODD to is the method which minimizes the maximum lateness KPI. Table 18 shows the comparisons. All the multiple rule methods shown assign EODD to at least one department. In the case of preferential attachment (PA), EODD is on the KPI frontier, so the multiple rule method shown is the next point along the frontier in the PA chart in Figure 18. The multiple rule method shown for each of the other routing distributions does better than EODD. However, the two performances are very close in all cases. They are certainly close enough for the confidence rectangles to overlap, so EODD may in fact out perform the best multiple rule method on some job sets. Moreover, the cost of the complexity of using multiple rules is unlikely to be compensated for by the performance difference.

Routing	Dept 1	Dept 2	Dept 3	Dept 4	Dept 5	Max Late	% Late
Empirical	SOPT	SOPT	SOPT	FIFO	EODD	49.54 weeks	66.95%
	EODD	-	-	-	-	49.54 weeks	68.88%
PA	EODD	-	-	-	-	24.94 weeks	45.81%
	EODD	EODD	EODD	EDD	EODD	25.06 weeks	45.74%
PF	EODD	EDD	EODD	EODD	EODD	13.36 weeks	26.35%
	EODD	-	-	-	-	13.36 weeks	26.37%
Triangle	EODD	EDD	EDD	EODD	EODD	8.76 weeks	20.35%
	EODD	-	-	-	-	8.82 weeks	20.41%
Uniform	EDD	EODD	EODD	FIFO	EDD	8.76 weeks	20.35%
	EODD	-	-	-	-	8.86 weeks	20.35%

Table 18: The maximum lateness minimizing methods compared to EODD for each routing distribution.

5.2.3. Comparing the number of rules

In this section, we compare the best performances of single rules and multiple rule methods using 2, 3 or 4 rules. It suffices to make the comparison graphically using charts in Figure 19. Each figure shows KPI frontiers for the best performing multiple rule methods grouped by the number of rules they employ: the green frontier is for methods with two rules, the purple for three rules, and the yellow for four rules. We include the KPI frontier for single rules in red for comparison. The analysis is the same in all five cases; the three multiple rule frontiers are not statistically distinguishable, as the rectangles of each colour in each figure entirely cover the frontiers of the other colours. Hence, regardless of a job-shops routing distribution, combining more than two rules is unlikely to result in performance gains.

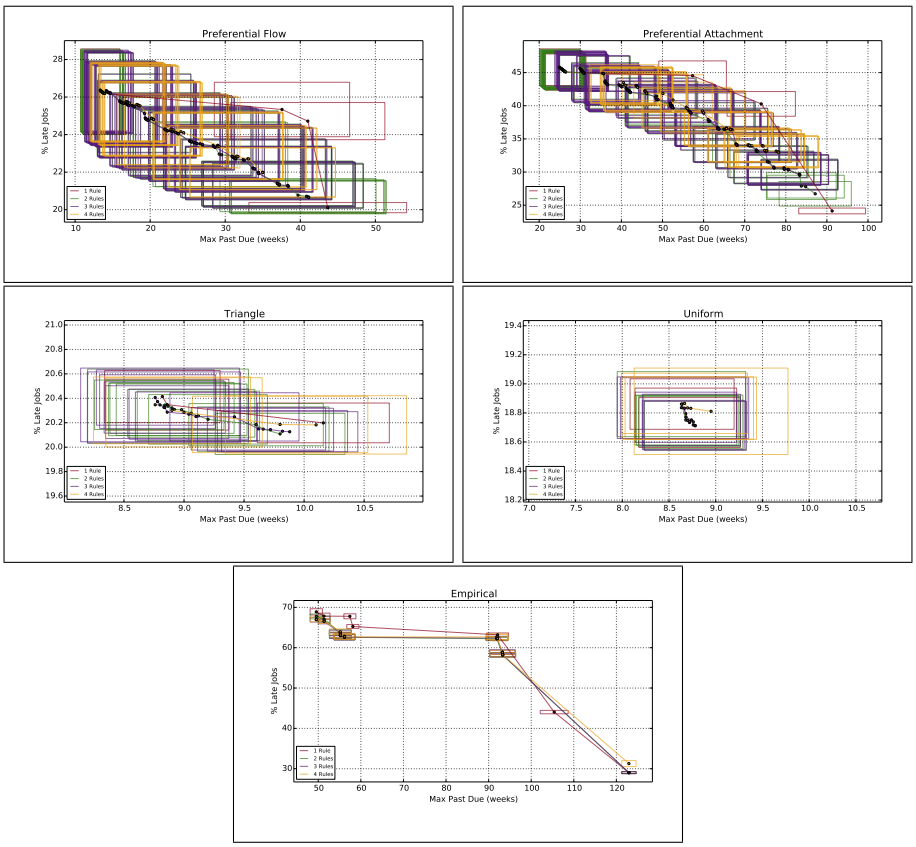

Figure 19: The KPI frontiers of simulations using 1, 2, 3 or 4 rule methods for each routing distribution.

6. Discussion

6.1. Just enough Complexity

Table 19 is a complete list of every dispatching rule which appears on one or more of the six KPI frontiers discussed in Sections 4 and 5. For each, the table shows the number of frontiers on which a given rule appears. Interestingly, the table only has nine rules listed, so despite the wide range of performance differences shown in Figure 11 between the industry partner data and the five routing distributions, a consistent set of dispatching rules appear on the frontiers. Moreover, the dispatching rules listed are among the simplest in concept and execution: EDD, SOPT, SNQ, FIFO, MRO, ERD all require a lookup and comparison; EODD and EFCD require calculations once per job and after that, it is a lookup and comparison; and LS requires two subtractions then compares the results for each execution. These nine rules performed better than any of dispatching rules which combine tardiness or job slack together with other rules in some way, all of which are somewhat more complicated in concept and execution.

Appendix A lists several complicated dispatching rules which select jobs for processing by considering all jobs in progress, not just those jobs in the queue of the machine on which the dispatching rule was triggered. These rules could potentially cause a machine m to wait for a high priority job to arrive in q_m instead of selecting a job to process immediately. However, preliminary experiments show these rules perform substantially worse than most of the rules included here.

It would seem that, at least some, simple dispatching rules have a performance advantage over the more complicated rules. It is worth noting that EODD, the best performing single rule, is not the simplest rule appearing on any of the KPI frontiers. Arguably, EDD is a simpler rule. The fact that EODD uses two crucial pieces of information for each job, the due date and the amount of work remaining, instead of just the due date, as EDD does, could explain its superior performance. As EODD is currently used by our industry partner, it would seem to be just complex enough.

Rule	Count	Rule	Count
EODD	6	EDD	6
SOPT	5	SNQ	3
FIFO	2	MRO	2
EFCD	1	ERD	1
LS	1		

Table 19: The number of times each rules appears on a KPI frontier for industry partner data or a routing distribution. EODD and EDD both achieve the maximum, as they appear on all six KPI frontiers.

6.2. KPI Trade-offs

Table 20 lists five performance comparisons for EODD and SOPT. The performances on the uniformly routed job sets are not listed because SOPT is not on that particular KPI frontier. For the rest, the two rules are on the extreme ends of each KPI frontier; EODD minimizes the maximum lateness KPI and SOPT minimizes the percentage of late jobs KPI.

Depting	Rule	Max Late	% Late
Industry	EODD	25.0 weeks	28.27%
	SOPT	87.0 weeks	13.78%
Empirical	EODD	49.54 weeks	68.88%
	SOPT	122.94 weeks	29.03%
PA	EODD	24.94 weeks	45.81%
	SOPT	91.28 weeks	24.13%
PF	EODD	13.36 weeks	26.37%
	SOPT	43.66 weeks	20.11%
Triangle	EODD	8.82 weeks	20.41%
	SOPT	10.16 weeks	20.20%

Table 20: EODD and SOPT performance comparisons for industry partner data and all but the uniform routing distribution.

Figure 20 offers an insight into the different ways these two rules process jobs. In the figure, each (x, y) point on the curves indicates that y% of jobs were incomplete at x weeks past their due dates; for instance, when using EODD on the industry partner data, roughly 12% of the jobs completed 10 weeks past due or more, though while using SOPT only 5% of jobs were completed at 10 weeks past due or later. The differences stem from the different ways EODD

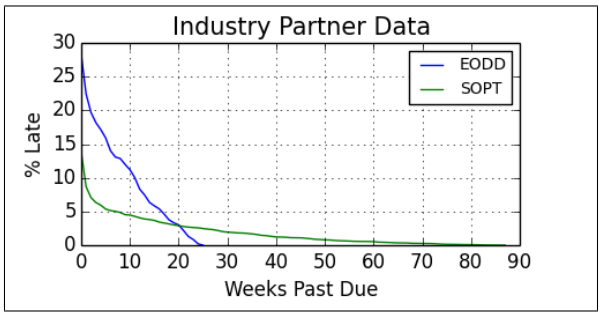

Figure 20: The percentage of late jobs which were incomplete at week n.

and SOPT select jobs for processing on the most utilized machines in a job-shop which we discuss in the following paragraphs. For that discussion, take m to be a heavily utilized machine in our industry partner's job-shop or a job-shop $\mathcal{J}$ with an empirical or a preferential attachment routing distribution.

For EODD, each operation of a newly released job is assigned an operation due date, and that date is determined in large part by the length of the operation: the longer the operation the earlier the date assigned. The due dates of operations preceding a long operations are even earlier. Hence, jobs with long operations are likely to be processed earlier than jobs without long operations. If the long operation is also processed on a heavily utilized machine m, many other jobs in q_m will wait for that long operation to complete, increasing the chances that those waiting jobs will be completed past due. These facets show up in the comparatively high percentage of late jobs in the performance of EODD regardless of routing distribution.

The figure shows that using SOPT ensures most of the jobs are completed by their due dates, some 86%, but a small number of jobs, about 2%, are completed at 30 weeks or more past due, some more than 80 weeks past due. The aim of SOPT is to increase the amount of parallel processing in a job-shop; by choosing the job with the shortest current operation, the rule SOPT selects the job which can be most quickly moved onto another machine for processing or completion.

However, jobs with one or more long operations processed on a heavily utilized machine such as m will be delayed, because a job with a shorter current operation is likely to enter q_m and be selected by the SOPT dispatching rule. That is until the all the jobs in the simulation have passed their release date, and the number of incomplete jobs drop off. For a real job-shop with the same routing distribution as $\mathcal{J}$, there are always new jobs being released. Hence, using SOPT in a strict way in a real job-shop amounts to having a small set of jobs which are never completed in exchange for having roughly half as many late jobs. While that reduction of late jobs is large, we learned that it would unreasonable for our industry partner to accept that 2% of jobs are left incomplete or refused, because leaving jobs incomplete or refusing them adversely affects company reputation (Industry-Partner, 2013).

6.3. Travel Times

In Section 3.4, we mentioned that in our simulations the travel time between machines is modeled as one day. This is the maximum time allowed by our industry partner in their job-shop. By choosing the maximum, we were justified in modeling the machinery and people which move the jobs as an infinite resource instead of a limited one, which greatly simplified our job-shop simulation. However, choosing the maximum and treating travel as an infinite resource has several consequences.

One of the consequences is that the rules which tend to get the most jobs traveling at once will generally out perform the others, as this maximally exploits the infinite resource. This is why SOPT out performs SPT for instance. SOPT selects the job which can be made to move between machines in the shortest amount of time, while SPT does not guarantee that.

Another consequence is due to the fact that the travel times are long compared to the total machine processing, excluding SUB, which had a few effects. First, rules, like EDD, ERD or FIFO, do not use travel times for their calculations, so they each "leave out" a significant factor from their dispatching logic. Second, several rules use a snapshot of the state of some or all machines in their calculations. The long travel time between machines often resulted in the snapshots being "out of date" by the time a dispatched job reached another machine, reducing the effectiveness of the dispatching logic. This was particularly true of the rules in Appendix A each of which used the state of other machines in quite complex calculations. SNQ may be the only rule we simulated which uses snapshots effectively, in that observing a short queue implies a higher chance of there being an empty queue after a job completes its movement.

The last consequence we discuss is that the long travel times may explain the success of EODD. We mentioned in Section 6.1 that EODD's dispatching logic considers two pieces of information: the due date and the operation processing times. In fact, the operation due dates implicitly consider the one day travel times because each operation due date is at least one day earlier than the next operations due date in sequence. Hence, the operation due date schedule proposed for each job is a tight schedule including the travel time.

6.4. Simplicity of One Rule

Several times in Sections 4 and 5, we mention that the best performing multiple rule scheduling methods do out perform EODD, but that the cost of the additional complexity of managing two or more dispatching rules in a real job-shop is greater than the potential gain regardless of the underlying routing distribution. In this section, we make our reasoning for this explicit.

In Sections 4 and 5, we calculated arc-elasticities, and the elasticities show that EODD assigned to every machine is the second best scheduling method for simultaneously minimizing the two KPIs we consider. The best method in every case, save for the preferential attachment job sets, is a multiple rule method. Table 21 shows a comparison between EODD and the best multiple rule method for each routing distribution; Table 21 is Table 18 with the Industry lines added to show the comparison for the industry partner data.

The first assertion we make for our argument is that the performance gains from using the multiple rule method over EODD are all small; the decreases in Table 21 are all less than 1%, save one which is 2.8%, which is indeed small. The next assertion is that employing more than one dispatching rule introduces complexity into a real-shop in several ways. This complexity comes primarily from the need to create at least two training streams, process maps, and sets of reports. Further, departments with different dispatching rules may have different priorities for the work in progress. Differing priorities can lead to a lack of flexibility due to difficulty with moving staff between departments, and worse, it could lead to disputed priorities between departments. While we

cannot quantify the costs of the second assertion, we contend that just the cost of replicating training, processes and reports is likely to exceed the the gains we see in Table 21. When coupled with the fact our industry partner already uses EODD, we are confident that the preceding argument is enough to justify our recommendation that our industry partner avoid using multiple rule scheduling methods.

Depting	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Max Late	% Late
Industry	EODD	EODD	EODD	EODD	SOPT	25.0 wks	27.93%
	EODD					25.0 wks	28.27%
Empirical	SOPT	SOPT	SOPT	FIFO	EODD	49.54 wks	66.95%
	EODD					49.54 wks	68.88%
PA	EODD					24.94 wks	45.81%
	EODD	EODD	EODD	EDD	EODD	25.06 wks	45.74%
PF	EODD	EDD	EODD	EODD	EODD	13.36 wks	26.35%
	EODD					13.36 wks	26.37%
Triangle	EODD	EDD	EDD	EODD	EODD	8.76 wks	20.35%
	EODD					8.82 wks	20.41%
Uniform	EDD	EODD	EODD	FIFO	EDD	8.76 wks	20.35%
	EODD					8.86 wks	20.35%

Table 21: The maximum lateness minimizing multiple rule methods compared to EODD.

6.5. Routing Distribution Difficulty and Control

In comparing the performances of dispatching rules on the routing distributions, we notice that particular routing distributions are more "difficult" to process than others. For instance, every scheduling method used on the uniformly routed job sets performed better by large factors in both KPIs over the best methods used on the preferential attachment job sets. The relative difficulty of two routing distributions can be ascertained by their relative positions of the KPI frontiers in Figures 11 and 17 in the previous section, and in general, we can say routing distribution A is more difficult to process than B if the KPI frontier of scheduling method performances for A is entirely above and to the right of the frontier for B. For the frontier comparison to be reasonable, the number of machines and the number of operations in the job sets must be the same or very similar, as is the case here.

The notion of routing distribution difficulty suggests that caution should be taken in selecting or creating job sets to model job-shops for algorithms and heuristics. The very large performance difference of every scheduling method indicates that the uniform routing job sets are poor models for our industry partner's job-shop. Given the complex and specialized nature of real job-shops, uniformly routed job-shops are very likely in the minority, so uniformly routed job sets are likely a poor model for most real manufacturing job-shops. Worryingly, of the ones considered here, the uniform routing distribution is the easiest

routing distribution to implement in terms of programming complexity, so it may be the routing distribution used for many simulations. If that is the case, the applicability of the simulated results is limited.

On the other hand, being able to compare the difficulties of two routing distribution suggests a mathematical language for another way to approach optimizing job-shop performance, which is to actively alter the routing distribution of a job-shop. Ideally, a job-shop would have a uniform routing distribution, which is to say that each machine performs roughly the same amount of work so there are no bottleneck machines. This ideal is typically impossible given the specialization of equipment and labour required for most forms of production. It may be possible to reduce the number of machines or the size of the bottlenecks by making strategic equipment purchases to redirect work from bottleneck machine. It might also be possible to explicitly include additional information in the job designing process, such as projected machine workloads, in an attempt to avoid further burdening bottleneck machines. Of course, these ideas to alter difficult routing distributions do not originate with us, but rather from having observed our industry partner. We only offer the idea of discussing these changes as attempts to control a job-shop's routing distribution.

7. Conclusion

7.1. Scheduling method Recommendation for our industry partner

The primary question of this research is whether our industry partner change or keep their current scheduling method. To answer this, we simulate the processing of two years worth of work in a job-shop model of our industry partner's factory using over 30000 different scheduling methods. The simulations are based on the set of jobs our industry partner actually processed in a two year time period.

All of our simulated scheduling methods employ job-shop dispatching rules because any method we recommend needs to be easy to implement, to teach and to use for our industry partner. The methods assign rules to departments, and the departments we used are the same as the departments our industry partner used at the time of our initial investigations.

The performances of all these methods were compared using two KPIs: the maximum number of weeks a job completed past its due date, and the percentage of jobs which completed late. The best methods appear on the KPI frontier, and the best rule among those rules on the frontier is the one which simultaneously minimizes both KPIs to the greatest degree.

In Section 4.2, we found the best method to be one which assigned the two dispatching rules EODD and SOPT. This method is on the KPI frontier for the industry partner data simulations, so it is one of the best performing methods for that production data.

Scheduling Method					Max Late	% Late
EODD	EODD	EODD	EODD	SOPT	25 weeks	27.93%
-	-	EODD	-	-	25 weeks	28.27%

Table 22: Best performing scheduling methods on the industry partner data.

However, we also note in that section that this multiple rule method performed similarly to the method employing only EODD on all machines. The two methods and their performances are listed again in Table 22. EODD is the second best method for simultaneously minimizing both KPIs. Hence, most of the benefits of using the two rule method can be achieved using EODD, which our industry partner already does. As such, we recommend the following:

> Our industry should make no change to their current scheduling method of using the earliest operation due date (EODD) dispatching rule on all machines.

7.2. Modeling a Job-Shop

We also aimed to extend our results to other job-shops by simulating using random production data. Very early on, we noticed that all of the scheduling methods were performing significantly better by a factor of at least two over the industry partner data on the random production data we had created at that point. This was despite the fact that nearly every property of the random data was sampled from the industry partner data. The missing piece was a model for how to construct a job route, the sequence of job-shop machines a job must be processed on before it is complete. The initial random production data we simulated was created using a uniform routing distribution, so we had inadvertently biased our dispatching rule performance.

A routing distribution is such a model, and we define and simulate five different routing distributions which we named the uniform, triangle, preferential flow, preferential attachment, and empirical routing distributions. The five routing distributions were used to create 250 different random job sets, fifty per routing distribution. These job sets were used in our job-shop simulator, and the average performance of each scheduling method was computed with 95% confidence intervals for the five routing distributions. Figure 17 shows the KPI frontiers for the five distributions along with the industry partner KPI frontier, and strikingly, there is no overlap of even the pictured confidence intervals. The figure also shows that the best scheduling methods for the uniform routing distribution significantly outperform the best methods on nearly every other routing distribution including the industry partner data.

We derive two conclusions from that error and results of simulating the routing distributions. The first is that dispatching rule performance is greatly affected by the routing distribution of the job-shop where the rules are employed, so the simulated performance of a given dispatching rule may only be applicable to job-shops with the same routing distribution as in the simulation. Moreover, the uniform routing distribution job sets are very different from our industry partner's production data. The difference is visible in Figure 17 which shows the two KPI frontiers together. Hence, any dispatching rule performance claims made using a uniform routing distribution are unlikely to be broadly applicable. Further, we conjecture that performance claims of any job-shop algorithm run on uniformly routed production data are not broadly applicable, and that any job-shop algorithm run on a uniformly routed job set will perform significantly worse when that algorithm is run on our industry partner's data.

The second conclusion is that EODD is a good scheduling method for our industry partner to use even if, as we suspect, their routing distribution changes over time. Throughout Section 5, we found that the scheduling method employing EODD on all machines performed better than nearly all other methods. The consistency of EODD's performance across the routing distributions we simulated is evidence that employing EODD is a good hedge against a changing routing distribution, as it seems likely that EODD will remain a top performing dispatching rule for a new routing distribution.

7.3. Future Work

We finish with a discussion of how this work might be extended.

7.3.1. Extending the job-shop model

For simplicity, we left several common features of job-shop models out of our model. Our scheduling method simulations might more accurately reflect reality if we also modeled machine breakdown, employee availability or job transit times to name a few.

We are also interested in modeling what we have called job shepherding. We observed that from time to time a group employees would "shepherd" a job through its processing; the group would ensure that processing began immediately upon arrival at a machine, processing was carried out by selected individuals, transit times were short, and all problems arising for the job were quickly solved. Modeling some or all of these techniques is a way to include in the job-shop model some of the positive impacts people working in the real job-shops can have on processing performance.

7.3.2. Extend to other job-shops

Our research was fundamentally based on production data from our industry partner, and we were uncommonly lucky to have had access to that. However, we do not know if the results we observed here are truly generalizable to other job-shops, even with our consideration of alternative routing distributions. For that, we would need similar access to more production data from one or more real job-shops. Even gaining access to similar data from our industry partner at later date would allow us to test whether their routing distribution changes significantly over time, as we suggest in the second conclusion of Section 7.2, and if so, we could test whether the analyses above also apply to that new production data.

7.3.3. Explore creating Departments

The performance gains we observed from using multiple dispatching rules were no significant enough for us to recommend to our industry partner to implement such scheduling methods. However, we did not exert any control over the departments themselves, and there may be significant performance gains to be had from strategically designing departments for use with dispatching rule scheduling methods. For example, it may be the case that some dispatching rules work well on those machine which typically process a job's first operation, but poorly on the finishing machines. In that case, the starting machines and finishing machines could each be grouped into their own departments, and assigned specialized dispatching rules.

7.3.4. Further investigate routing distributions

Several ideas occur to extend the research into routing distributions. The first potential extension is the conjecture made in the first conclusion of Section 7.2. We conjecture that the routing distribution will affect all job-shop algorithms as significantly as we have observed it affecting dispatching rules. Proving the conjecture could amount to modeling the various job sets as mixed integer programs, and finding the optimal solutions.

Another extension would be to create a routing distribution which better models our industry partner's production data. One way to show we did not achieve this here is by using all the dispatching rules on a KPI frontier as a kind of signature for features of the underlying routing distributions. All of the frontiers in Figure 17 are different from the industry partner data (IPD) frontier, so when viewed as routing distribution signatures, we see that none of the routing distributions we defined are very similar to the IPD. The routing distributions preferential attachment and flow both have a parameter, a_d and f_d, which may be adjusted so that these routing distributions might be made to better resemble the IPD.

The empirical routing distribution also needs to be "fixed;" its frontier is the furtherest from the IPD frontier of all the frontiers shown in Figure 21, a figure similar to Figure 11. We attempted to make a fix to the empirical routing distribution by determining processing times by using unit processing times; the jobs our industry partner works on are primarily batch jobs consisting of one to many units, so to determine an operation's processing time, we sampled a set of unit processing times and multiplied by the sampled number units for a job. The result is shown as the 'Empirical II' frontier in Figure 21. In the figure, the 'Empirical II' frontier is similar to the 'Empirical' frontier, so this particular attempt to better model the IPD failed. It is clear that additional research is required to determine the key features of industry data which must be replicated in the creation of similar random data.

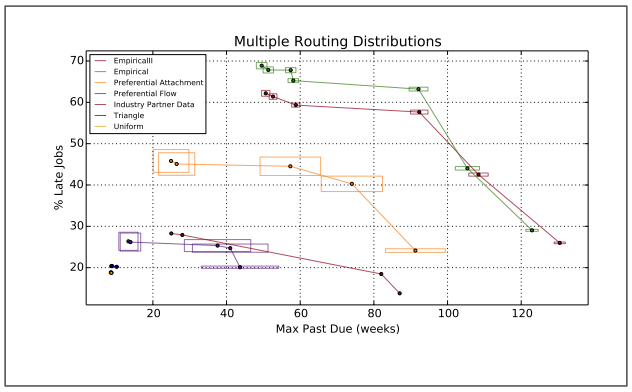

Figure 21: The single rule KPI frontiers for each of the six routing distribution considered above, and an additional frontier for a second empirical routing distribution.

Bibliography

Adams, J., Balas, E., Zawack, D., 1988. The shifting bottleneck procedure for job shop scheduling. Management science 34 (3), 391–401.

Armentano, V. A., Scrich, C. R. a., 2000. Tabu search for minimizing total tardiness in a job shop. International Journal of Production Economics 63 (2), 131–140.

Barabási, A., Albert, R., Oct. 1999. Emergence of scaling in random networks. science 286 (5439), 509–512.

CMC, 2012. Manufacturing Our Future: A Manufacturing Action Plan for Canada. Tech. rep., Canadian Manufacturing Coalition.

Colorni, A., Dorigo, M., 1994. Ant system for job-shop scheduling. Belgian Journal of Operations Research, Statistics and Computer Science 34 (1), 39–53.

De Giovanni, L., Pezzella, F., 2010. An Improved Genetic Algorithm for the Distributed and Flexible Job-shop Scheduling problem. European Journal of Operational Research 200 (2), 395–408.

Garey, M., Johnson, D., Sethi, R., 1976. The complexity of flowshop and jobshop scheduling. Mathematics of operations research 1 (2), 117–129.

Goren, S., Sabuncuoglu, I., Koc, U., 2012. Optimization of schedule stability and efficiency under processing time variability and random machine breakdowns in a job shop environment. Naval Research Logistics 59 (1), 26–38.

Holstein, W., Berry, W., 1970. Work flow structure: An analysis for planning and control. Management Science 16 (6), B324–B336.

Huang, K., Liao, C., Apr. 2008. Ant colony optimization combined with taboo search for the job shop scheduling problem. Computers & Operations Research 35 (4), 1030–1046.

Industry-Partner, 2013. private communication with a manager from our industry partner.

Kaban, a. K., Othman, Z., Rohmah, D. S., 2012. Comparison of dispatching rules in job-shop scheduling problem using simulation: a case study. International Journal of Simulation Modelling 11 (3), 129–140.

Lunsdorf, O., Muller, K., Scherfke, S., Vignaux, T., 2012. SimPy: Event discrete simulation for Python.
URL `http://simpy.readthedocs.org/`

Manne, A., 1960. On the job-shop scheduling problem. Operations Research 8 (2), 219–223.

McKay, K., Safayeni, F., Buzacott, J., 1988. Job-shop scheduling theory: what is relevant? Interfaces 18 (4), 84–90.

Nowicki, E., Smutnicki, C., 2005. An Advanced Tabu Search Algorithm for the Job Shop Problem. Journal of Scheduling 8 (2), 145–159.

Nuijten, W., Aarts, E., 1996. A computational study of constraint satisfaction for multiple capacitated job shop scheduling. European Journal of Operational Research 7 (95), 269–284.

Pan, Q.-K., Tasgetiren, M. F., Liang, Y.-C., 2008. A discrete differential evolution algorithm for the permutation flowshop scheduling problem. Computers & Industrial Engineering 55 (4), 795–816.

Panwalkar, S., Iskander, W., 1977. A survey of scheduling rules. Operations Research 25 (1), 45–61.

Philipoom, P., Fry, T., 1990. The robustness of selected job-shop dispatching rules with respect to load balance and work-flow structure. Journal of the Operational Research Society 41 (10), 897–906.

Pinedo, M., 2009. Planning and scheduling in manufacturing and services. Springer.

Ruiz, R., Vázquez-Rodríguez, J. A., 2010. The hybrid flow shop scheduling problem. European Journal of Operational Research 205 (1), 1–18.

Sculli, D., Tsang, K., 1990. Priority dispatching rules in a fabrication/assembly shop. Mathematical and Computer Modelling 13 (3), 73–79.

Sha, D., Hsu, C.-Y., 2006. A hybrid particle swarm optimization for job shop scheduling problem. Computers & Industrial Engineering 51 (4), 791–808.

Shah, P., 2004. SELECTING A MIX OF DISPATCHING RULES. Ph.D. thesis, Ryerson University.

Shmoys, D. B., Stein, C., Wein, J., 1994. Improved Approximation Algorithms for Shop Scheduling Problems. SIAM Journal on Computing 23 (3), 617–632.

Tavakkoli-Moghaddam, R., Khalili, M., Naderi, B., 2008. A hybridization of simulated annealing and electromagnetic-like mechanism for job shop problems with machine availability and sequence-dependent setup times to minimize total weighted tardiness. Soft Computing 13 (10), 995–1006.

Vaessens, R. J. M., Aarts, E., Lenstra, J., 1994. Job shop scheduling by local search. INFORMS JOURNAL ON COMPUTING 8, 302–317.

Vela, C. R., Varela, R., González, M. a., 2008. Local search and genetic algorithm for the job shop scheduling problem with sequence dependent setup times. Journal of Heuristics 16 (2), 139–165.

Wei-ling, W., Jing, Y., 2013. A hybrid differential evolution algorithm for job shop scheduling problem to minimize the total weighted tardiness. Management Science and Engineering (ICMSE), 2013 International Conference on, 294–300.

Yu, H., Liang, W., Apr. 2001. Neural network and genetic algorithm-based hybrid approach to expanded job-shop scheduling. Computers & Industrial Engineering 39 (3-4), 337–356.

Zhang, R., Wu, C., 2010. A divide-and-conquer strategy with particle swarm optimization for the job shop scheduling problem. Engineering Optimization 42 (7), 641–670.

Zhang, R., Wu, C., 2011. A simulated annealing algorithm based on block properties for the job shop scheduling problem with total weighted tardiness objective. Computers & Operations Research 38 (5), 854–867.

Appendices

A. Failed Dispatching Rules

In this appendix, we describe two dispatching rules which we did not use in our final simulations. In the initial tests, both rules, when compared to rules on the KPI frontiers, performed very poorly. Further, the run-time was greatly increased due to each of these rules selecting a highest priority job from all the work in progress, not just the jobs in a particular queue. We describe them here for completeness.

Both these rules use the concept of a job ranking. A job ranking is a function $R : \mathcal{J} \to \mathbb{R}$. The jobs can be weakly ordered or prioritized by rank. Each of the dispatching rules in Section 3.5 exploits a job ranking to select a job for processing; for instance, the rule EDD selects the job with the earliest due date which can be considered as the number of days since January 1, 2000, a real number. Since we required our dispatching rules to be implementable with a list, we can find a mapping of jobs to real numbers for each dispatching rule which can be used as a job ranking and the highest ranked job is the one the dispatching rule selects for processing.

A.1. Minimum Completion Time

The aim of this rule is to predict the completion time of a job based on a the priority of a job as determined by the ranking function R, and opportunistically, complete the jobs which can be completed the earliest. The idea is to calculate the maximum amount of time a job is expected to wait at all the machines it has yet to be processed on, and estimate a completion time as the sum of the current time, t_c; the remaining processing time and the remaining transit time, w_j^{rem}; and that maximum wait time:

$$comp(j) = t_c + w_j^{rem} + waittime(j).$$

Calculating the wait time, as we conceive it, is straight forward once some notation has been established.

Let m_i be the machine which processes the ith operation of j. We calculate two quantities: the arrival time, a_{ij}, of j in the queue of m_i, and the start time, s_{ij}, of operation o_{ij}.

- The arrival time a_{ij} is calculated by assuming j will not wait in any future queue, and a_{ij} is equal to t_c plus all the processing and transit time required before j is moved into q_{m_i}. It is zero if o_{ij} is complete or j is already in q_{m_i}.

- The start time s_{ij} is an estimate of the time that j is waiting while higher priority jobs are processed on m_i. Hence, s_{ij} is zero if o_{ij} is complete, and otherwise, s_{ij} is calculated as the sum of the current time plus the processing times of all current operation of jobs $k \in q_{m_i}$ where $R(k) \geq R(j)$.

With that, the wait time of j is the maximum difference between the start times and arrival times:

$$waittime(j) = \max\{0, s_{1j} - a_{1j}, s_{2j} - a_{2j}, \ldots, s_{\ell j} - a_{\ell j}\}.$$

The job with the minimum $comp(j)$ is selected for processing.

The calculation of $waittime(j)$ takes advantage of the fact that jobs with higher priority than j will be processed whether j is in the same queue or not, so if no jobs with rank greater than or equal to $R(j)$ enter q_{m_i}, then $s_{ij} - a_{ij}$ decreases steadily to zero regardless of whether j is in q_{m_i} or not. Of course, that assumption is not reasonable, particularly if the job ranking is dynamic, so the wait time of a job could vary wildly even while it is stationary in a queue. Whats more, the job ranking function is called many more times than for our other dispatching rules, as we repeatedly calculate the rank of every job which may interact with j.

A.2. Highest Priority WIP

The idea of this rule is to determine whether a high priority job, j^*, is soon to arrive at a machine m, and if so, select a job already in q_m to process which will be complete before j^* arrives in q_m. A job ranking function R is required to determine job priority. To implement this rule, we need to introduce two pieces of notation. The first is a list, ℓ_m, maintained at each machine m, and the list has all of the released jobs which still need to be processed on m. The second is $w_j^{c \to m}$, which is the amount of processing and transit time required for j to be moved from its current location to q_m.

B. Python Code

All code is stored in the GitHub repository `https://github.com/salvor7/JobshopSimulation`

B.1. Simulation Code

The simulation code files are *simulator.py*, *timer.py* and *sleep.py*.

B.2. Dispatching Rule Code

The dispatching rule definition python code is specified in *ranking_rules.py*.

Algorithm 3 High Priority WIP Dispatching Rule

Require: Machine m where rule is triggered.
Require: The list of jobs ℓ_m.

procedure DISPATCHING RULE
    ETA $= \infty$.
    $L = \ell_m$.
    while $L \neq \emptyset$ **do**
        Find the highest priority job, j^*, listed on L.
        if $j^* \in q_m$ and $p_j^c <$ ETA **then**
            return j^* for processing and Stop.
        else
            Remove j^* from L.
            if ETA $> w_j^{c \to m}$ **then**
                Set ETA $= w_{j^*}^{c \to m}$
    return Sleep Signal.

B.3. Routing Distribution Code

The python code defining routing functions used to create job sets is cont *create_data_files.py*.

C. KPI frontiers

The KPI frontier file is stored in the GitHub repository `https://github.com/salvor7/JobshopSimulation` as *multirulefront.csv*.

Index