# Determining the Characteristic of Difficult Job Shop Scheduling Instances for a Heuristic Solution Method

Helga Ingimundardottir and Thomas Philip Runarsson

School of Engineering and Natural Sciences, University of Iceland
{hei2,tpr}@hi.is

**Abstract.** Many heuristic methods have been proposed for the job-shop scheduling problem. Different solution methodologies outperform other depending on the particular problem instance under consideration. Therefore, one is interested in knowing how the instances differ in structure and determine when a particular heuristic solution is likely to fail and explore in further detail the causes. In order to achieve this, we seek to characterise features for different difficulties. Preliminary experiments show there are different significant features that distinguish between easy and hard JSSP problem, and that they vary throughout the scheduling process. The insight attained by investigating the relationship between problem structure and heuristic performance can undoubtedly lead to better heuristic design that is tailored to the data distribution under consideration.

## 1 Introduction

Hand crafting heuristics for NP-hard problems is a time-consuming trial and error process, requiring inductive reasoning or problem specific insights from their human designers. Furthermore, within a problems class, such as job-shop scheduling, it is possible to construct problem instances where one heuristic would outperform another. Depending on the underlying data distribution, different heuristics perform differently, commonly known as the *no free lunch* theorem [1]. The success of a heuristic is how it manages to deal with and manipulate the characteristics of its given problem instance. So in order to understand more fully how a heuristic will eventually perform, one needs to look into what kind of problem instances are being introduced to the system. What defines a problem instance, e.g. what are its key features? And how can they help with designing better heuristics?

In investigating the relationship between problem structure and heuristic effectiveness one can research what [2] calls *footprints* in instance space, which is an indicator how an algorithm generalises over the instance space. This sort of investigation has also been referred to as *landmarking* [3]. It is evident from experiments performed in [2] that one-algorithm-for-all problem instances is not ideal. An algorithm may be favoured for its best overall performance, however

Y. Hamadi and M. Schoenauer (Eds.): LION 6, LNCS 7219, pp. 408–412, 2012.

it was rarely the best algorithm available over various subspaces of the instance space. Thus when comparing different algorithms one needs to explore how they perform w.r.t. the instance space, i.e. their footprint.

In this study, the same problem generator is used to create 1,500 problem instances, however the experimental study in section 3 shows that MWRM works well/poorly on a subset of the instances. Since the problem instances are only defined by processing times and its permutation, the interaction between the two is important, because it introduces hidden properties in the data structure making it easy or hard to schedule with for the given algorithm. These underlying characteristics or features define its data structure. So a sophisticated way of discretising the instance space is grouping together problem instances that show the same kind of feature behaviour, in order to infer what is the feature behaviour between *good* and *bad* schedules.

It is interesting to know if the difference in the structure of the schedule is time dependent, is there a clear time of divergence within the scheduling process? Moreover, investigation of how sensitive is the difference between two sets of features, e.g. can two schedules with similar feature values yield completely contradictory outcomes, i.e. one poor and one good schedule? Or will they more or less follow the same path? This essentially answers the question of whether is is in fact feasible to discriminate between *good* and *bad* schedules using the currently selected features as a measure. If results are contradictory, it is an indicator the features selected are not robust enough to capture the essence of the data structure. Additionally, there is also the question of how can one define 'similar' schedules, what measures should be used? This paper describes some preliminary experiments with the aim of investigating the feasibility of finding distinguishing features corresponding to *good* and *bad* schedules in JSSP.

Instead of searching through a large set of algorithms (creating an algorithm portfolio) and determining which algorithm is the most suitable for a given subset of the instance space, as is generally the focus in the current literature [4,5,2], our focus is rather on a single algorithm and understanding *how* it works on the instance space – in the hopes of being able to extrapolate where it excels in order to aid its failing aspects.

The outline of the paper is as follows, in section 2 priority dispatch rules for the JSSP problem are discussed, what features are of interest and how data is generated. A preliminary experimental study is presented in section 3. The paper concludes with a summary of main findings and points to future work.

## 2 Job-Shop scheduling

The job-shop scheduling task considered here is where $n$ jobs are scheduled on a set of $m$ machines, subject to the constraint that each job must follow a predefined machine order and that a machine can handle at most one job at a time. The objective is to schedule the jobs so as to minimize the maximum completion times, also known as the makespan. For a mathematical formulation of JSSP the reader is recommended [6].

**Table 1.** Feature space $\mathcal{F}$ for JSSP. Features 1–13 can vary throughout the scheduling process w.r.t. tasks that can be dispatched next, however features 14–16 are static

| $\phi$ | Feature description |
|---|---|
| $\phi_1$ | processing time for job on machine |
| $\phi_2$ | start-time |
| $\phi_3$ | end-time |
| $\phi_4$ | when machine is next free |
| $\phi_5$ | current makespan |
| $\phi_6$ | work remaining |
| $\phi_7$ | most work remaining |
| $\phi_8$ | slack time for this particular machine |
| $\phi_9$ | slack time for all machines |
| $\phi_{10}$ | slack time weighted w.r.t. number of operations already assigned |
| $\phi_{11}$ | time job had to wait |
| $\phi_{12}$ | size of slot created by assignment |
| $\phi_{13}$ | total processing time for job |
| $\phi_{14}$ | total processing time for all jobs |
| $\phi_{15}$ | mean processing time for all jobs |
| $\phi_{16}$ | range of processing times over all jobs |

### 2.1 Single-Priority Dispatching Heuristic

Dispatching rules are of a construction heuristics, where one starts with an empty schedule and adds on one job at a time. When a machine is free the dispatching rule inspects the waiting jobs and selects the job with the highest priority. A survey of more than 100 of such priority rules was given in 1977 by [7]. In this paper however, only most work remaining (MWRM) dispatching rule will be investigated.

In order to apply a dispatching rule a number of features of the schedule being built must be computed. The features of particular interest were obtained from inspecting the aforementioned single priority-based dispatching rules. The temporal scheduling features applied in this paper are given in Table 1. These are not the only possible set of features, they are however built on the work published in [6,4] and deemed successful in capturing the essence of a JSSP data structure.

### 2.2 Data Generation

Problem instances were generated stochastically by fixing the number of jobs and machines and sampling a discrete processing time from the uniform distribution $U(1, 200)$. The machine order is a random permutation of $\{1, ..., m\}$. A total of 1,500 instances were generated for a six job and six machine job-shop problem.

In the experimental study the performance of the MWRM, $\mu_{\text{MWRM}}$, and compared with its optimal makespan, $\mu_{\text{opt}}$. Since the optimal makespan varies between problem instances the following performance measure is used:

$$\rho = \frac{\mu_{\text{MWRM}}}{\mu_{\text{opt}}}. \tag{1}$$

## 3 Experimental Study

In order to differentiate between problems, a threshold of a $\rho < 1.1$ and $\rho > 1.3$ was used to classify *easy* and *hard* problems. Of the 1500 instances created, 271 and 161 problems were classified *easy* and *hard* , respectively.

**Table 2.** Features for *easy* and *hard* problems are drawn from the same data distribution (denoted by ·)

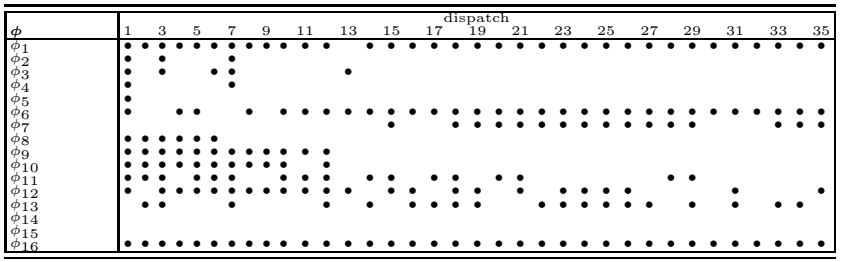
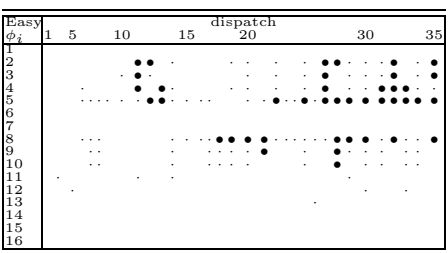



**Table 3.** Significant correlation (denoted by ·) for *easy* (left) and *hard* (right) problems and resulting ratio from optimality, $\rho$ defined by (1). Commonly significant features across the tables are denoted by •.



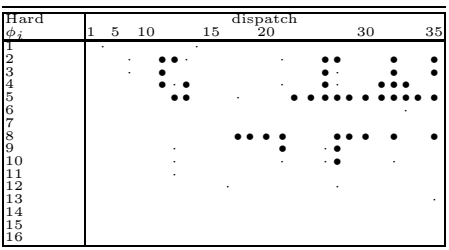




Table 2 reports where data distributions are the same (denoted by ·). From the table one can see that distribution for $\phi_1$, $\phi_6$, $\phi_{12}$ and $\phi_{16}$) are (more or less) the same throughout the scheduling process. However there is a clear time of divergence for distribution of slacks; step 6 for $\phi_8$ and step 12 for $\phi_9$ and $\phi_{10}$.

In order to find defining characteristics for *easy* and *hard* problems, a (linear) correlation was computed between features (on a step-by-step basis) to the resulting ratio from optimality. Significant features are reported in Table 3 for *easy* and *hard* problems, (denoted by ·). As one can see from the tables, the significant features for the different difficulties are varying. Some are commonly significant features across the tables (denoted by •).

## 4 Discussion and Conclusion

From the experimental study it is apparent that features have different correlation with the resulting schedule depending in what stage it is in the scheduling process, implying that their influence varies throughout the scheduling process. And features constant throughout the scheduling process are not correlated with

the end-result. There are some common features for both difficulties considered which define JSSP on a whole. However the significant features are quite different across the two difficulties, implying there is a clear difference in their data structure. The amount of significant features were considerably more for easy problems, indicating their key elements had been found. However, the features distinguishing hard problems were scarce. Most likely due to their more complex data structure their key features are of a more composite nature.

The feature attributes need to be based on statistical or theoretical grounds. Thus scrutiny in understanding the nature of problem instances is of paramount importance in feature engineering for learning. Which yields feedback into what features are important to devout more attention to, i.e. features that result in a failing algorithm. In general, this sort of investigation can undoubtedly be used in better algorithm design which is more equipped to deal with varying problem instances and tailor to individual problem instance's needs, i.e. a footprint-oriented algorithm.

Although this methodology was only implemented on a simple single-priority dispatching rule heuristic, the methodology is easily adaptable for more complex algorithms. The main objective of this work is to illustrate the interaction of a specific algorithm on a given problem structure and its properties.

## References


1. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1), 67–82 (1997)
2. Corne, D.W., Reynolds, A.P.: Optimisation and Generalisation: Footprints in Instance Space. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI, Part I. LNCS, vol. 6238, pp. 22–31. Springer, Heidelberg (2010)
3. Pfahringer, B., Bensusan, H.: Meta-learning by landmarking various learning algorithms. In: Machine Learning (2000)
4. Smith-Miles, K.A., James, R.J.W., Giffin, J.W., Tu, Y.: A Knowledge Discovery Approach to Understanding Relationships between Scheduling Problem Structure and Heuristic Performance. In: Stützle, T. (ed.) LION 3. LNCS, vol. 5851, pp. 89–103. Springer, Heidelberg (2009)
5. Smith-Miles, K., Lopes, L.: Generalising Algorithm Performance in Instance Space: A Timetabling Case Study. In: Coello Coello, C.A. (ed.) LION 5. LNCS, vol. 6683, pp. 524–538. Springer, Heidelberg (2011)
6. Ingimundardottir, H., Runarsson, T.P.: Supervised Learning Linear Priority Dispatch Rules for Job-Shop Scheduling. In: Coello Coello, C.A. (ed.) LION 5. LNCS, vol. 6683, pp. 263–277. Springer, Heidelberg (2011)
7. Panwalkar, S., Iskander, W.: A Survey of Scheduling Rules. Operations Research 25(1), 45–61 (1977)