Protecting a Sensitive Queue from Arrival Variability

Mathieu Vandenberghe*, Stijn De Vuyst*, El-Houssaine Aghezzaf*, Herwig Bruneel†
*Department of Industrial Systems Engineering and Product Design, Ghent University
Technologiepark 903, 9052 Zwijnaarde, Belgium
†Department of Telecommunication and Information Processing, Ghent University
St-Pietersnieuwstraat 41, 9000 Gent, Belgium
mathieu.vandenberghe@ugent.be

***Abstract*—Minimizing item waiting time between stages is a general focus of operations research, and of particular concern for certain industries. We propose a two-stage production system where, to minimize the waiting time before stage 2, we focus on spreading the completion times of the stage 1 machines across the available interval. We contrast this objective with a similar problem defined in a healthcare context, but that has an assumption of fixed assignment. We obtain insights in the added value that free assignment can provide, by comparing the solutions of a local search method for assignment, with those of a reference case where assignment is fixed. Computational results show that this added value is highest in cases where task means differ insufficiently to be ordered effectively, and where task distributions have low variance. For the discussed instances, significant reductions in item waiting times can be achieved while making minimal concessions on expected makespan.**

***Index Terms*—Manufacturing Systems, Uncertain Service Times, Waiting Time, Local Search**

I. INTRODUCTION

Many industrial and information systems can be viewed as a chain of interlinked subsystems. A chief focus of operations research is to optimize these links, particularly the process of transferring items from one stage to the next. Ideally such a transfer would be seamless and predictable, as variability in the rate of arrival can lead to problems with scheduling, inventory and manpower: all of which are likely to lead to delays and loss of efficiency. In some sectors (notably the food industry) it is furthermore critical to minimize the total (waiting) time of items between production stages, for example because of regulations that limit the exposure time of food to ambient temperature. In that sense, the queues in such a production line can be called 'sensitive'.

We envision a two-stage production system (represented in Fig. 1) that handles exactly M items. Stage 1 consists of K parallel machines whose processed items must all be served by the machine in stage 2 and which has a single sensitive queue. We assume that all task durations and service times are independent, although task durations (stage 1) can have distinct distributions while service times (stage 2) are identical for all items. In stage 1, the items assigned to any of the parallel machines are processed contiguously on that machine: when one item is completed, the processing time of the next starts immediately. As it is in our interest to minimize the waiting time of items in the sensitive queue, we should avoid 'burstiness' at the entrance of the queue.

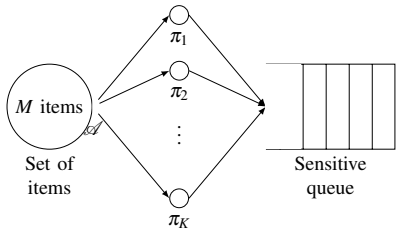

Fig. 1: The model represented as a queuing system.

We can use various methods to minimize this burstiness, especially if we are willing to focus on it at the expense of all other metrics (e.g. we could assign all items to a single machine, and delay processing until we are sure stage 2 is free). In practice however, minimizing the waiting time of items must be done in tandem with other considerations, such as minimizing makespan and maintaining a good takt time.

Therefore we restrict ourselves to a straightforward strategy to reduce burstiness: within an acceptable makespan, our objective will be to spread the completion times of stage 1 as uniformly as possible over this makespan. Note that if we were to also penalize the under-utilization of stage 2, a better strategy would likely be to let the completion times resemble the 'dome-shaped' pattern common in appointment scheduling [1]. Given the complexity of the problem however, this paper focuses on creating a uniform spread of completion times as a first approximation.

The resulting objective has strong similarities with the Break-In-moment (BIM) Problem [2] in operating room scheduling. Its motivation is that hospitals must deal with the unexpected arrivals of urgent emergency patients. Free rooms are typically not available, and so the earliest possibility for the emergency to 'break into' the schedule is when any room finishes its surgery. This leads BIM to have a novel scheduling objective: minimizing the expected maximum time between these potential 'break-in-moments'. We have focused on solving the more general case where surgery durations are stochastic but emergency arrivals are unknown: the Stochastic Break-In-Moment (SBIM) Problem [3].

Minimizing this worst-case risk is equivalent to trying to create a schedule where completion times are uniformly spread. This means we should be able to re-use our solution methods [3] in the context of the sensitive queue process we outlined above. However there is one major difference: the standard assumption for SBIM is that tasks are pre-assigned to machines. In a hospital context this assumption is defensible, as it is common practice to assign operating room blocks to various specialities as part of tactical planning [4]. Yet in many industrial processes this assumption is unwarranted, and both the assignment $\mathscr{A}$ of tasks to machines in stage 1, *and* their internal order π could be changed.

The free assignment of tasks adds a significant factor of complexity, but the added freedom should allow us to reach better objective values than were possible in the SBIM case. We are interested in quantifying this improvement (the 'value of free assignment'), and particularly under what *conditions* this improvement justifies the extra computational cost.

II. LITERATURE

Scheduling problems, i.e. the general goal of scheduling M jobs on K machines in a way that optimizes an objective function, was one of the first Operations Research topics to be studied and remains an active research area to this day. The classification scheme of [5] identifies three major classes based on the characteristics of the machine environment and the jobs: dedicated machines, parallel machines, and the various 'shop' problems (open shop, flow shop, job shop). The problem we study falls under parallel machines, as each task needs to be processed only once by any of the machines.

Within scheduling, various objective functions have been researched; with the most common one (across various classifications) being makespan minimization [6]. Problems related to makespan minimization can also be integrated with various extensions such as maintenance [7], availability constraints [8], deteriorating jobs [9].

Our particular objective function is rather unique, as the waiting time of individual items in the production system is mainly a concern for specific applications (such as the food industry). The objective of 'total weighted tardiness' is related, but this refers to the delay of items versus a predefined due date (set by management or the customer). Indeed the literature focused on reducing item waiting times (and more generally placing demands on each of the completion times) is comparatively more limited. One major exception is the goal of minimizing completion time variance (CTV), which was first discussed by [10] and has received much attention. For instance, [11] seeks to minimize the CTV of items in both a single-machine and parallel machine system. The authors use a genetic algorithm to establish an upper bound, and then analyze the performance of various heuristics. In most applications however, CTV is focused on minimizing the variance of each task's average expected completion time (whatever that might be); it does not try to spread the completion times in a particular way. Finally, the closest analogue to our objective is the very relevant contribution by [12] , who also focus on creating uniformity of output across a given interval (by reducing CTV). However, the authors focus on flow shop scheduling, where each item must be processed by several stages of a single machine each.

Our model also considers stochasticity. [13] summarizes the recent objectives in the literature for job shop scheduling under uncertainty, the majority of which are again related to either makespan or tardiness. Three broad strategies exist for incorporating stochasticity. First, one can seek to protect the schedule by minimizing the probability that certain (large) deviations occur; commonly referred to as 'robustness'. Second, if specific assumptions can be about the type of stochasticity, this can allow for analytically calculating the effect of a particular schedule change.

We will be using a third approach, known as Sample Average Approximation (SAA), and studied in detail by [14]. SAA consists of creating and optimizing across a large amount of scenarios based on the input distributions. The advantage of this approach is that it is quite general, and can be extended to a variety of input distributions. In the context of unrelated parallel machines, [15] creates artificial scenarios to analyze the maximal regret (in terms of makespan) any solution can result in. [16] proposes two exact algorithms and several heuristics for a very similar makespan minimization objective, but for identical parallel machines.

The rest of this paper is organized as follows: section 3 formally defines the SBIM problem with assignment, as well as its objective function. Section 4 discusses the solution methods used, and the reference method by which to compare the SBIM problem with free assignment to the version with fixed assignment. In section 5 we analyze the computational results, and section 6 concludes the paper.

III. THE ASSIGNMENT OBJECTIVE

Formally, the SBIM problem with free assignment consists of M tasks with independent stochastic durations $\mathbf{P} = (P_1, \ldots, P_M)$ which must be scheduled on K machines. An assignment matrix $\mathscr{A} = (A^1, \ldots, A^K) \in A$ determines which tasks are assigned to which machines, and a global schedule $\pi = (\pi^1, \ldots, \pi^K) \in \Pi$ determines their sequence on each machine. Any combination of assignment $\mathscr{A}$ and schedule π will lead to a set of completion times $C_i, \quad i \in \mathscr{I} = \{1, \ldots, M\}$. We now define the BIM sequence B_i as the completion times C_i in ascending order. The intervals between the BIMs form the Break-In-Intervals (BIIs) $S_i = B_i - B_{i-1}$, $B_0 = 0$, $i \in \mathscr{I}$. The objective is to find a schedule π that minimizes the expected value of the largest BII length:

$$v^* = \min_{\substack{\pi \in \Pi \\ \mathscr{A} \in A}} g(\mathscr{A}, \pi), \quad \text{with} \quad g(\mathscr{A}, \pi) = \mathrm{E}[\max_{i \in \mathscr{I}} S_i(\mathscr{A}, \pi, \mathbf{P})] \quad (1)$$

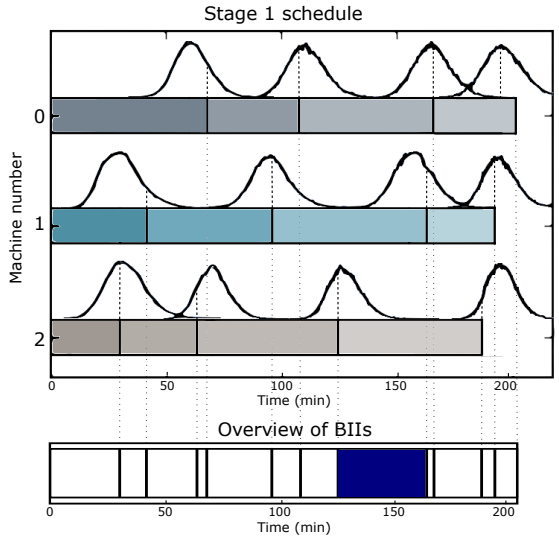

Fig. 2: Empirical densities of all BIMs for an instance $K = 3$ and $M = 12$ (all lognormal distributions). Task durations and associated BIMs represent one particular scenario, with the largest BII in blue.

and where the expectation is taken over the joint distribution of $\mathbf{P}$. The optimal value v^* in (1) is the maximum difference between adjacent completion times. Given this objective, the best strategy was to spread the break-in-moments as uniformly as possible across the session interval. Fig. 2 provides a visual example of the SBIM problem.

In order to estimate the value of a particular combination of assignment and schedule $(\pi, \mathscr{A})$, we use the most general method of creating a large set of N scenarios, each of which contains a sampled value for each of the task duration distributions. The associated sample average approximation (SAA) objective across N scenarios will then be:

$$\hat{g}_N(\mathscr{A}, \pi) = \frac{1}{N} \sum_{n=1}^{N} \max_{i \in \mathscr{I}} S_i(\mathscr{A}, \pi, \mathbf{P}_n,), \tag{2}$$

As this objective must be evaluated across N scenarios, and cannot be decomposed by machine, it is fairly expensive. Nonetheless, it allows for the use of various heuristics.

IV. SBIM SOLUTION METHODS

A. Value of the Assignment Solution

To quantify the potential improvement for the SBIM objective that can be achieved by allowing free assignment of tasks to machines, it is clear that for any instance, we will seek to obtain the assignment (and internal ordering) that minimizes the objective value. However when contrasting this value to a reference case where assignment is fixed (and only internal ordering is possible), an important decision is *which fixed assignment* should be used to create a valid comparison.

As covered in section II, a majority of the literature on production scheduling problems is preoccupied with minimizing the expected makespan. We thus assume that when not optimizing the assignment for the purposes of the SBIM objective, a likely assignment to be used in factory settings would be one that minimizes the expected makespan. The internal ordering of this fixed assignment would not impact the makespan however, and so it could still be optimized with the SBIM objective in mind. This would yield the reference value for the SBIM with fixed assignment. Our goal is to show the further improvement in item waiting times that can be achieved by also optimizing the assignment; but we will do so *while not exceeding* the minimized expected makespan by a measure of α. This ensures that expected makespan remains a chief consideration.

While it is not within the scope of this paper to provide solution methods that solve the SBIM with free assignment to optimality, we nonetheless need solution methods that can make a good estimate of the objective value. In the remainder of this section we specify the main solution methods we chose, and their scheduling operators.

B. Scheduling operators

Local search methods are iterative algorithms; in each iteration j, they restrict themselves to searching a specific neighborhood of the current solution $(\mathscr{A}, \pi)^{(j)}$, and select the most promising neighbor as the next iteration's start $(\mathscr{A}, \pi)^{(j+1)}$. Each iteration's search is driven by one of three scheduling operators: OpOrder, OpSwap, or Op2for1. OpSwap swaps two tasks on different machines, and OpOrder swaps tasks on the same machine (internal ordering). For the former, the neighborhood is defined as the combination of all schedules that could be reached by swapping any two tasks in the current solution, provided they are on different machines. Neighborhood for the latter works identically, except it swaps any two tasks on the same machine.

As for Op2for1, its purpose is to change the number of tasks assigned to each machine. It finds sets of tasks $\{A, B, C\}$ where A and B are scheduled on the same machine, and where the sum of their means $(\mu_A + \mu_B)$, is close to the mean of C; i.e. $(1-\gamma)\mu_C \leq \mu_A + \mu_B \leq (1+\gamma)\mu_C$. It then moves tasks A and B to the machine on which C is scheduled; and vice versa. Its neighborhood contains as many neighbors as there are sets of tasks $\{A, B, C\}$ for which this holds. The value for γ used in our experiments is $\gamma = 0.20$.

C. Tabu Search with assignment and ordering operators

Tabu Search [17] is a well-known local search method that iteratively explores the neighborhood of promising solutions. To avoid getting stuck in local optima, it uses a mechanism known as a tabu list: the algorithm keeps a list of previous solutions (or at least a property of them) and does not return to them. The algorithm stops after a certain number of total

iterations, or after a smaller number of iterations wherein no improvements on the objective value were found.

In our implementation, each iteration j consists of selecting one of the three scheduling operators (according to predefined probabilities) and constructing a neighborhood around $(\mathscr{A},\pi)^{(j)}$. We then search the entire neighborhood, and choose the neighbor with the best objective value, but excluding (i) solutions already in the tabu list and (ii) solutions that increase the expected makespan by more than α. The best neighbor will be the starting point for iteration $j+1$, and is added to the tabu list. Whenever we find a new best objective value, we clear the tabu list.

For the instances discussed in section V, we use operator probabilities $\{P_{\text{OpOrder}} = 0.70, P_{\text{OpSwap}} = 0.25, P_{\text{Op2for1}} = 0.05,\}$. As a stop condition, we terminate the algorithm after 500 iterations have occurred without an improvement to the current best solution, or after 2000 iterations in total.

D. Reference assignment: makespan Tabu Search

To benchmark the case of SBIM with fixed assignment, we first find an assignment of tasks to machines that minimizes the expected makespan. We then look for the internal ordering that would still minimize the SBIM objective.

The initial makespan optimization is solved with a Tabu Search, using the two assignment operators (OpSwap and Op2for1) with probabilities $\{P_{\text{OpSwap}} = 0.80, P_{\text{Op2for1}} = 0.20,\}$. The subsequent internal ordering also consists of a Tabu Search, but using only the OpOrder operator. Both stages terminate based on the same criteria: after 1000 iterations in total, or after 500 iterations wherein no improvement on the current best solution was found.

An important note is that since the Tabu Search for internal ordering has a much smaller solution space to traverse (see section III), giving it the same amount of iterations in total gives it a significant advantage. Therefore our results are very unlikely to overstate the value of the assignment method.

V. COMPUTATIONAL RESULTS

To identify situations in which optimizing the assignment of tasks to machines will have a significant impact, we constructed various datasets. Each is characterized by a combination of 'Range' (the range covered by the means μ of the task distributions) and 'Var' (the size of the standard deviation σ). Table I details the nine resulting datasets.

We first compare the solution methods outlined above across each of the datasets. For the sake of readability we employ shorthand phrases for the solution methods: 'Only order (Tabu)' refers to the reference solution method (see section IV-D), where assignment is fixed. 'Tabu Assign & Tabu Order' refers to the Tabu Search with multiple operators (see section IV-C).

Table II lists the average maximum BII that results from each method's final solution, for each of the datasets. We also provide a 95% confidence interval, which we emphasize is for the maximum BII, not for the *average* maximum BII.

TABLE I: Datasets employed for experiments

	K	M	Parameters of normal distributions (min)
Range1Var1	6	38	$\mu = 70\ldots110$; $\sigma = 2\ldots6$
Range1Var2	6	38	$\mu = 70\ldots110$; $\sigma = 3\ldots12$
Range1Var3	6	38	$\mu = 70\ldots110$; $\sigma = 5\ldots25$
Range2Var1	6	38	$\mu = 65\ldots140$; $\sigma = 2\ldots6$
Range2Var2	6	38	$\mu = 65\ldots140$; $\sigma = 3\ldots12$
Range2Var3	6	38	$\mu = 65\ldots140$; $\sigma = 5\ldots30$
Range3Var1	6	38	$\mu = 50\ldots180$; $\sigma = 2\ldots6$
Range3Var2	6	38	$\mu = 50\ldots180$; $\sigma = 3\ldots12$
Range3Var3	6	38	$\mu = 50\ldots180$; $\sigma = 5\ldots30$

TABLE II: Performance results for $\alpha = 0.10$ across datasets.

	Only order (Tabu)		Tabu Assign & Tabu Order	
	Avg. max BII	Conf. Int.	Avg. max BII	Conf. Int.
Range1Var1	47.52	36.98 - 57.28	31.46	25.515 - 39.055
Range1Var2	42.66	31.52 - 56.58	32.263	27.74 - 44.40
Range1Var3	42.25	32.44 - 54.49	40.25	31.15 - 52.00
Range2Var1	43.49	32.67 - 53.87	28.55	23.51 - 35.26
Range2Var2	38.63	29.61 - 49.96	33.40	26.48 - 42.53
Range2Var3	42.01	31.88 - 54.76	40.42	30.98 - 52.52
Range3Var1	31.97	26.18 - 38.74	28.96	23.96 - 35.52
Range3Var2	37.22	29.93 - 46.75	32.451	26.47 - 40.37
Range3Var3	40.92	32.19 - 51.74	38.55	30.39 - 48.74

As expected, our results show that the SBIM with free assignment is able to find significantly better solutions than the version with fixed assignment. The Tabu Search for both assignment and ordering (IV-D) outperforms its counterpart on every instance.

The extent of the improvement varies however. Most notably, an increase in the variance of the task distributions reduces the added value of free assignment. This has an intuitive explanation: greater task uncertainty reduces the impact that even a good assignment can have. Consequently, high-variance instances improve their objective by only about 5%; low-variance instances increase by 10-30%.

The range of distribution means present in the dataset, also has an impact. For equal levels of variance, datasets with a larger range are able to obtain lower objective values. However the added value of optimizing assignment for these datasets, versus only optimizing internal order, is smaller. If there is a wide range of distribution means in the dataset, a search for the best internal order (within a fixed assignment) can already create very good schedules. On the other hand, when the range of distribution means is small, it becomes more critical to assign tasks to machines wisely; doing so can significantly decrease the average maximum BII.

We were also interested in how compatible our objective (focused on item waiting times) is with respect to that of makespan minimization. As the fixed assignment used in comparisons is one that minimizes expected makespan, we can surmise that other (free) assignments will have a worse

TABLE III: Performance results for the method 'Tabu Assign and Tabu Order', for various values of α.

	Avg. max BII for values of α			
	$\alpha = 0.20$	$\alpha = 0.10$	$\alpha = 0.05$	$\alpha = 0.01$
Range1Var1	31.75	31.46	31.51	33.41
Range1Var2	35.11	35.24	35.42	36.30
Range1Var3	40.19	40.25	40.30	40.57
Range2Var1	29.00	28.55	29.76	30.44
Range2Var2	32.88	33.40	34.01	34.46
Range2Var3	40.58	40.42	40.55	40.35
Range3Var1	28.18	28.96	28.51	28.81
Range3Var2	32.37	32.45	33.03	33.91
Range3Var3	38.51	38.55	39.12	39.06

makespan. The factor α limits this however; e.g. $\alpha = 0.10$ means that new assignments can increase makespan by at most 10%. To this end a further experiment is shown in Table III, where we applied the solution method with assignment (Tabu-assign and Tabu-order) to all datasets for various values of α.

As we can see, a more restrictive value for α does worsen the SBIM objective. The effect is not very significant however, and does not meaningfully change the conclusions resulting from Table II. After all, the definition of the SBIM objective function punishes large BIIs in general. This means it is generally not in the interest of solution methods to deviate too far from the best makespan, as a larger makespan means a larger interval across which to spread completion times. In this sense the two objective functions are aligned.

Even when setting a constraint that limits makespan increase to $\alpha = 0.01$, we still find solutions that retain 75% of the gains over the fixed assignment results in Table II ($\alpha = 0.10$). We conclude that for the instances we analyzed, optimizing the SBIM objective function can be done *in addition* to finding an excellent makespan, and does not significantly interfere with it.

VI. CONCLUSION

We outlined a two-stage production system composed of a set of parallel machines as stage 1, and a sensitive queue as stage 2. If reducing the waiting times of items exiting stage 1 is a concern, a possible strategy is to uniformly spread completion times. This creates the same objective function as used in the Stochastic Break-In-Moment problem, but without the assumption that the assignment of tasks to machines is already fixed.

To assess the possible improvement in objective value, we proposed two local search methods to find good solutions. We then tested them on datasets whose task distributions varied in two ways: the range of the various distribution means present, as well as the variance of each distribution. Results show that allowing for free assignment of tasks to machines has the greatest impact in instances where the variance of the input distributions is relatively low, and where there is not much variety in tasks means. In all but the highest-variance instances we analyzed, the objective value improvement was at least 10%. Furthermore, it could be obtained even when making minimal changes to the expected makespan.

To our knowledge this is the first work that applies the SBIM objective function in an environment where free assignment is possible. In this application, we have however restricted ourselves to the goal of spreading completion times as uniformly as possible, as we were only concerned with reducing the waiting times of items themselves. If we extend our concern to the under-utilization of the sensitive queue as well, this trade-off will demand a more sophisticated spread of the completion times, which will in turn require new algorithms. Further work is possible in this area.

REFERENCES

[1] R. Hassin and S. Mendel. "Scheduling Arrivals to Queues: A Single-Server Model with No-Shows". *Management Science* 54.3 (2008), pp. 565–572.

[2] J. T. van Essen et al. "Minimizing the waiting time for emergency surgery". *Operations Research for Health Care* 1.2-3 (2012), pp. 34–44.

[3] M. Vandenberghe et al. "Surgery Sequencing to Minimize the Worst-Case Waiting Time of Emergent Patients" (2018). Unpublished.

[4] E. Hans, M. Van Houdenhoven, and P. J. Hulshof. "A Framework for Health Care Planning and Control". *Handbook of Healthcare System Scheduling* 168 (2012), pp. 303–320.

[5] P. Brucker. *Scheduling Algorithms*. Vol. 93. 2007, pp. 1–52.

[6] Q. K. Pan and R. Ruiz. "Review of flowshop heuristics (Ruiz 1995)". *Computers and Operations Research* 40.1 (2013), pp. 117–128.

[7] E. H. Aghezzaf, A. Khatab, and P. L. Tam. "Optimizing production and imperfect preventive maintenance planning's integration in failure-prone manufacturing systems". *Reliability Engineering and System Safety* 145 (2016), pp. 190–198.

[8] C.-Y. Lee. "Machine scheduling with an availability constraint". *Journal of Global Optimization* 9.3 (Dec. 1996), pp. 395–416.

[9] J. N. Gupta and S. K. Gupta. "Single facility scheduling with nonlinear processing times". *Computers and Industrial Engineering* 14.4 (1988), pp. 387–393.

[10] A. G. Merten and M. E. Muller. "Variance Minimization in Single Machine Sequencing Problems". *Management Science* 18.9 (1972), pp. 518–528.

[11] R. Rajkanth, C. Rajendran, and H. Ziegler. "Heuristics to minimize the completion time variance of jobs on a single machine and on identical parallel machines". *International Journal of Advanced Manufacturing Technology* (2016), pp. 1–14.

[12] R. Leisten and C. Rajendran. "Variability of completion time differences in permutation flow shop scheduling". 54 (Feb. 2015).

[13] X. Hao et al. "Effective multiobjective EDA for bi-criteria stochastic job-shop scheduling problem". *Journal of Intelligent Manufacturing* 28.3 (2017), pp. 833–845.

[14] A. J. Kleywegt, A. Shapiro, and T. Homem-de-Mello. "Sample Average Approximation Method for Stochastic Discrete Optimization". *SIAM Journal on Optimization* 12.2 (2002), pp. 479–502.

[15] W. Naji, M.-l. Espinouse, and V.-d. Cung. "Modelling, Computation and Optimization in Information Systems and Management Sciences". 360 (2015), pp. 343–355.

[16] X. Xu et al. "Robust makespan minimisation in identical parallel machine scheduling problem with interval data". *International Journal of Production Research* 51.12 (2013), pp. 3532–3548.

[17] F. Glover. "Tabu Search - Part I". *ORSA journal on Computing* 2 1.3 (1989), pp. 4–32.