22nd International Congress on Modelling and Simulation, Hobart, Tasmania, Australia, 3 to 8 December 2017
mssanz.org.au/modsim2017

Scheduling a Production Line Using Heuristics and Constraint Programming

R. García-Flores [a], A. Schutt [ab] and R. Batres [c]

[a]*CSIRO Data61, Locked Bag 38004, Docklands, Victoria, 8012*
[b]*The University of Melbourne, Parkville, Victoria, 3010*
[c]*ITESM Campus Cuernavaca, Autopista del Sol Km. 104 + 060, Col. Real del Puente Xochitepec 62790, Morelos, Mexico*
Email: Rodolfo.Garcia-Flores@csiro.au

Abstract: Sequencing and scheduling are activities that are of paramount importance in the man-ufacturing industries. In today's economic environment, the efficiency gains produced by effective scheduling represent not only a competitive advantage, but also are crucial to keep customers happy through adequate service levels and to ensure adequate use of resources. In this paper, we minimise the makespan in a single-machine environment with release dates and sequence-dependent setup times, or a $1|s_{jk}, r_j|C_{max}$ scheduling problem in the notation introduced by Lawler et al. [1982]. This problem is based on a deodorant manufacturing facility where a number of products are produced, and a sequence must be determined such that the costs of switching from one family of products to another are minimised, subject to raw material availability and delivery dates. Schedules must be produced for a horizon of two weeks, which normally requires planning the production of around forty products (from a total of around three hundred) which may belong to any one of nine product families. The major component of the cost associated with changing from one family of products to another is due to cleaning the line when products from different families are produced in sequence. We present results obtained with two solution methodologies: first, we use the common strategy of producing a good initial solution via dispatching rules and then refining this solution through Tabu Search, and second, we develop a Constraint Programming model. Our results show that finely-tuned heuristic and Constraint Programming models can produce a satisfactory result with realistic data sets in a short execution time.

Keywords: *Production scheduling, constraint programming, Tabu search, dispatching rules*

1. INTRODUCTION

Optimal scheduling of production is an essential tool used by manufacturing companies to provide accurate, real-time production plans, support critical business decisions, and ensure that produced quantities are available on the promised dates. Planning tools incorporate data from forecasts, production projections and estimates, backlogs, availability of material, existing capacities, management goals and policies, and even operators' experience. Optimal schedules not only grant competitive advantage, but are also crucial to keep customers happy by achieving adequate performance measures.

In this paper, we address a one-machine scheduling problem with sequence-dependent setup times, release dates and due dates, which is known to be NP-hard Allahverdi et al. [1999]. This problem is based on a deodorant manufacturing facility where a number of products are produced, and a sequence must be determined such that the costs of switching from one family of products to another are minimised, subject to raw material availability and delivery dates. More specifically, the problem is to determine the optimal production sequence of individual products (also known as *variants*), such that the costs of switching the families of products to which the variants belong (also called *formats*) are minimised, subject to raw material availability and due dates. The costs of switching families comprise the cost of wasted materials, services, staff and, most importantly, the cost of cleaning the line by using a silicone-based cleaning fluid, which is a costly process. Currently, the schedule is obtained by hand by a human planner, in a process that is time-consuming and that does not guarantee that all the material availability constraints are being respected. To solve the problem, we test two methodologies. First, we use the standard two-step approach of obtaining a seed solution via a composite dispatching rule in the first stage, and then using an improvement-heuristic method in the second stage to refine the initial solution and arrive at a near-optimal solution. The second solution method is based on a Constraint Programming model.

In scheduling, a *dispatching rule* is a heuristic that suggests a job ordering in an attempt to optimise a simple objective. A *composite dispatching rule* is the combination of elementary dispatching rules, designed to satisfy more complex objectives than the elementary rules alone. Work on the development of specific dispatching rules and composites is extensive, as well as in the development of methodologies to fine-tune the parameters of these rules. However, comprehensive surveys of existing dispatching rules are few and far between. Examples of these include Blackstone et al. [1982] and Holthaus and Rajendran [1997]. Specifically, the scheduling rule we apply is the Apparent Tardiness Cost with Setups, or ATCS, which was first introduced in Lee et al. [1997]. The ATCS rule was developed to solve the problem of minimising the *total weighted tardiness* (the tardiness of a job is the maximum between zero and the difference between the completion date and the due date of that job) on a single machine in the presence of sequence-dependent setup times, and has been applied successfully to industrial scheduling systems (e.g., by Lamothe et al. [2012] for the pharmaceutical and cosmetics industry). In Lee and Pinedo [1997], the values of the parameters were determined through extensive experiments, and Chen et al. [2010] proposed an improved methodology of parameter calculation based on face-centred cube experimental designs, as introduced by Montgomery [2005].

In the present paper, we use the ATCS rule to calculate a seed solution schedule. Dispatching rules build complete schedules and are of the *constructive* type, as opposed to heuristics of the *improvement* type, used to refine existing schedules, usually through *local search* [Pinedo, 2010]. These local search heuristics attempt to find a schedule in the *neighbourhood* of the current schedule that is better than the current schedule. The heuristic accepts or rejects a candidate solution as the next schedule to move to, based on a given acceptance criterion. Examples of improvement heuristics include Tabu Search (which will be used in the present paper), Simulated Annealing, and Genetic Algorithms, whose pseudocode can be found in Brownlee [2010]. Finally, the alternative methodology considered in the present paper, Constraint Programming (CP) is considered to be effective for scheduling problems, particularly resource-constrained scheduling problems, or other combinatorial problems for which the integer programming model tends to be large or have a weak continuous relaxation. This is particularly true if the goal is to find a feasible solution or to minimise *makespan*, i.e., the last completion time [Bockmayr and Hooker, 2005].

In what follows, we describe the scheduling problem; then we explain the details of the methodologies used. Next, we present the results with data sets that closely resemble the situation in the deodorant manufacturing plant, and present a comparison. Finally, we discuss in the conclusions the practical

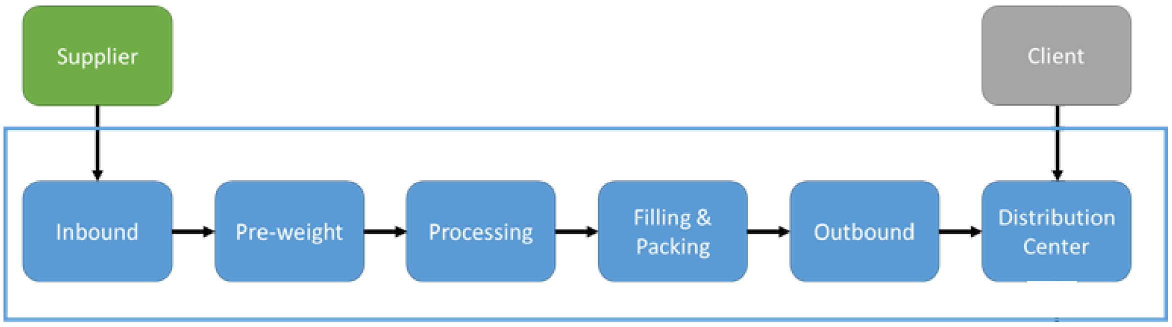


Figure 1. Process diagram showing all stages of production. This figure describes the flow of materials through the line, and not the scheduling problem, which is concerned only with the *Processing* stage.

implications of our work and list future improvements.

2. THE SCHEDULING PROBLEM

The schematic of the deodorant production line in Figure 1 shows the downstream and upstream stages of *processing*. Raw materials arrive from the *suppliers* at expected dates and times and are received as *inbound* materials, although a supplier may sometimes incur in a delay. The materials are *pre-weighted* and fed into the *process*, after which the deodorant tins are *filled and packed* under pressure in a sealed chamber. Changing the feed to the process between products within the same family is trivial, but switching between products of different families is an expensive process, as it requires cleaning the line by using a silicone-based cleaning fluid, in addition to the cost of wasted materials, services, and staff. Once packaged, the products are *bound* together, boxed and sent to *distribution centres*, from where they are shipped to the clients. *Clients* expect their product to be delivered before the agreed due dates. The plant produces nearly 300 individual products, and a sequence must be determined such that the costs of switching from one family of products to another are minimised, subject to raw material availability and delivery dates. The number of products per family varies from 9 to 85, from a set of nine families. Currently, the production schedule is calculated by hand by a human planner every Thursday. On Sunday, the plan is reviewed, so production can follow the calculated plan starting Monday. However, there is no certainty that the plan calculated manually respects all the constraints, especially those related to material availability. The production plan takes a few hours to calculate by hand and is used by different departments to manage the flows of materials inside the plant, and therefore all users should be able to interpret it easily. Ideally, the production plan should be calculated for a time horizon of two weeks, twice as long as the current manual plan, which normally requires planning the production of around forty products. Formally, the production system described above is a $1|s_{jk},\ r_j|C_{\max}$ scheduling problem, that is, the minimisation of the makespan in a single-machine environment with release dates, due dates and sequence-dependent setup times. Next section describes the methodology used to tackle the problem.

3. METHODOLOGY

In this section, we first describe an improvement strategy for a schedule, which is obtained using a composite dispatching rule, and second, we describe a Constraint Programming model.

3.1. Heuristic approach

The general approach consisted of 1) Using of a composite dispatching rule to obtain a seed solution, and 2) using of a heuristic method to arrive to an optimal or near-optimal solution.

Obtaining a seed solution. The composite rule we used was ATCS [Lee et al., 1997], developed for the $1|s_{jk}|\sum_j w_j T_j$ problem. According to the ATCS rule, the ranking index of job j at time t

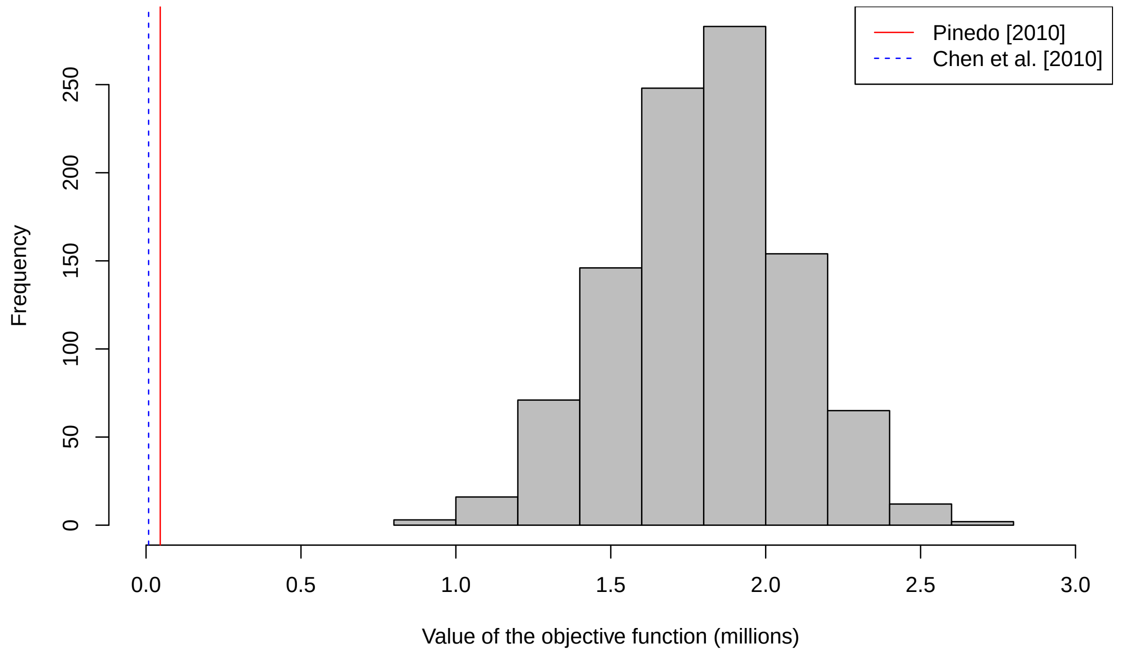


Figure 2. Histogram of objective function values for 1000 random permutations and the value of a sequence produced with the ATCS rule for a random problem of 40 jobs.

given that job j is processed after job l is calculated by

$$I_{\text{ATCS}}(t,j,l) = \frac{w_j}{p_j} e^{-\frac{\max(d_j - p_j - t)}{K_1 \bar{p}}} \; e^{-\frac{s_{lj}}{K_2 \bar{s}}} \,, \tag{1}$$

where w_j, p_j and d_j represent the weight, processing time, and due date of job j, respectively; $\bar{p}$ and $\bar{s}$ represent the average processing time and average setup time, respectively; s_{lj} is the sequence-dependent setup when job j is processed after job l, t is the current time; and K_1 and K_2 are scaling parameters. Chen et al. [2010] provide an extensive discussion on the importance of using the right values of the scaling parameters, and provide a general methodology based on mixture experiments to fit the scaling parameters. In addition to being a very insightful guide on the construction and use of composite dispatching rules, this article provides a set of parameters K_1 and K_2 that seems to outperform the calculation method given in Pinedo [2010]. To test the effectiveness of the ATCS rule with Chen et al.'s [2010] parameters, we produced R[1] code to generate parameters for random $1|s_{jk}|\sum_j w_j T_j$ problems. For one of these problems with 40 jobs, we produced 1000 permutations and plotted the resulting histogram in Figure 2. We tested the ATCS rule as reported in Pinedo [2010] and shown as a red, vertical line in Figure 2 and compared its value (45806) with the improved scaling parameters $K_1 = 1.391$ and $K_2 = 0.667$ reported in Chen et al. [2010] (the blue line in Figure 2 with tardiness value of 8188). It is clear that, for the artificial problem data used, the ATCS rule with Chen et al.'s [2010] parameters produces a seed with a smaller tardiness than the original rule.

Refining the solution. Tabu Search (TS) is a local search meta-heuristic that uses a list of "forbidden" moves to avoid getting stuck in local minima, and was first introduced in Glover and McMillan [1986].

[1]http://https://www.r-project.org/, accessed on the 7 of June 2017.

3.2. Constraint Programming

CP is a solving technology that encapsulates structures of a problem in constraints and actively uses specialised deduction algorithms for each constraint to reduce the search space. CP allows for high-level modelling without losing too much structure from the original problem and it is easy to add and/or remove constraints from the model, which makes it very flexible. CP models exploit the fact that all jobs must be executed in sequence. The decision variables in the array sequence determine which job is executed at first, second, and so forth. The variables in the arrays `startTimes` and `completionTimes` determine at what time the i-th processed job starts and ends, respectively. For each of these arrays, the constraint `all_different` ensures that each variable in the array takes a different value. All other problem constraints are modelled in a straightforward manner by linear constraints. However, the most important constraints for an efficient model are the two symmetry-breaking constraints, which significantly restrict the search space. The first symmetry-breaking constraint forces the first job to start immediately after the initial setup time or its release date if the initial setup time is earlier. The second symmetry-breaking constraint enforces that the i-th processed job immediately starts either its release date or at the end time of the previous job plus setup time. Both constraints force that a job is "left-shifted" when it has a slack in the schedule. This does not only allow better propagation, but it is also sensitive towards the objective, which benefits by earlier completion times. Note that the symmetry-breaking constraints might be invalid for other objectives.

4. RESULTS AND DISCUSSION

The heuristic approach was implemented in version 1.8 of the Clojure[2] language in a 64-bit Intel Xeon CPU with two processors of eight cores (2.27 GHz) each and 8 GB of RAM. The CP model was implemented in MiniZinc version 2.1.5 and solved using Chuffed[3]. All experiments were carried out in Ubuntu 16.04. The data sets were produced from a simulation in the R language to ensure that there exists at least one feasible solution assuming that the setup times between jobs of different families follow a normal distribution of $N(65.3, 29.38)$ minutes, the setup times between jobs of the same family follow a $N(1.41, 0.26)$ distribution, and that the amount to be produced is $N(10000, 3000)$ boxes, so that the processing time is calculated using the line speed. The gap window to define the release dates and due dates before start and after completion of the jobs in the simulation is set *ex post facto* to a mean of six hours and a standard deviation of two hours. For the heuristic, penalties for violating release and due dates are assigned randomly and normalised by the number of jobs so they add up to 100. The planning horizons tested ran from from two to 14 days.

Table 1 shows a summary of the near-optimal makespans and calculation times obtained by Chuffed using the `search4` search strategy; we call them near-optimal because, for CP, all parameters were rounded to the closest minute. The execution time in all cases is under 20 seconds.

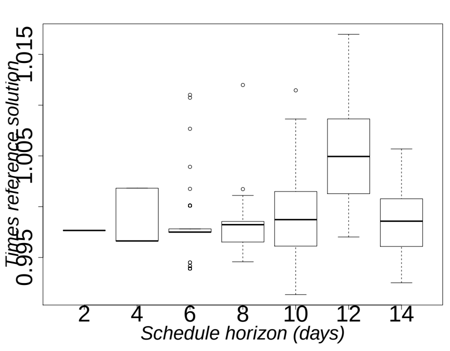
Table 1. Optimal makespan in minutes an execution time in seconds for MiniZinc model using Chuffed solver.

Horizon	**2 days**	**4 days**	**6 days**	**8 days**	**10 days**	**12 days**	**14 days**
No. of jobs	11	23	34	46	57	69	80
Makespan	3633	7469	10694	13229	19519	22371	25373
Execution time	0.021	0.175	1.308	7.194	4.563	16.73	2.496

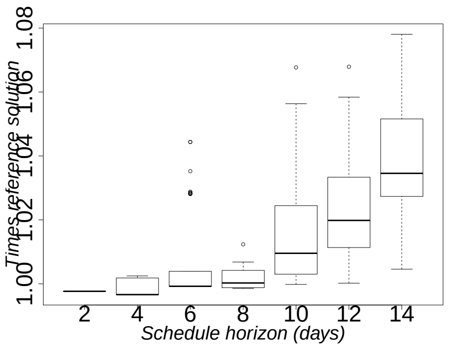
Figures 3 and 4 show the results of running the heuristic 50 times on each horizon using a Tabu list size of 100 and a neighbourhood size of 100. For the heuristic, the fitness function was defined as the makespan plus the weighted violations of release and due dates; note that the latter are incorporated as constraints in the CP model, not as part of the objective function. Similarly, the heuristic penalises with a very large sequence-dependent setup time all consecutive occurrences of products belonging to incompatible product families, whereas the CP model incorporates this requirement as a constraint.

[2]http://clojure.org/, accessed on the 7 of June 2017.

[3]https://github.com/chuffed/chuffed, accessed on the 7 of July 2017.

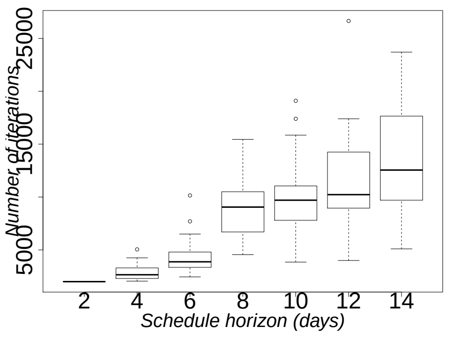


(a) Normalised makespans

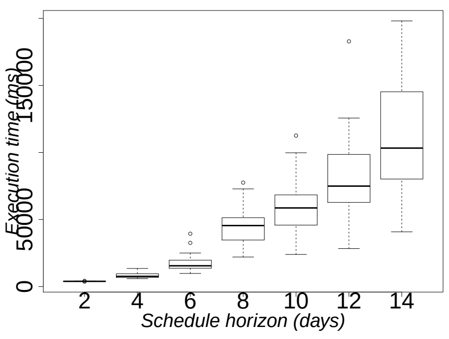


(b) Normalised fitnesses

Figure 3. Box plots of makespan and fitnesses running the heuristic 50 times on each horizon, normalised to the makespan of the sequence obtained by MiniZinc's Chuffed solver.



(a) Number of iterations



(b) Run times

Figure 4. Box plots of number of iterations and execution times running the heuristic 50 times on each horizon.

The stopping condition was defined as reaching 2000 iterations without improvement, and a sequence in the neighbourhood was obtained by swapping up to 3 pairs of jobs.

Figures 3a and 3b show normalised makespans and fitnesses respect to the near-optimal makespans obtained by Chuffed. Most makespans are lower than the optimal makespan because the penalty terms assigned to the violation of release and due dates are not high enough, and thus do not compensate when the heuristic finds a schedule with a lower-than-optimal makespan. This is the case for all planning horizons except 12 days. In the shortest time horizons, both fitness and makespans are below the solution found by Chuffed; this is because the heuristic uses the actual, fractional processing and setup times whereas the parameters used in CP are rounded to the nearest integer, and the problem is small enough for the heuristic to find the real optimum. In longer scheduling horizons, the fitness value found by TS diverges from the value found by CP with the number of jobs. Figures 4a and 4b show that the computational effort to obtain a heuristic solution increases with the planning horizon. On one hand, the CP model is much faster, although it involves more tuning than a general improvement heuristic like TS. On the other hand, the worst solution time with the heuristic in our experiments was 3.302 minutes, which is acceptable for an industrial application of the type presented.

5. CONCLUSIONS AND FURTHER WORK

We have presented and compared two solution methods for a $1|s_{jk},\ r_j|C_{\max}$ scheduling problem: a Tabu Search heuristic and a Constraint Programming model. The analysed problem is based on an actual manufacturing facility where a number of products are produced, and a sequence must be determined such that the costs of switching from one family of products to another are minimised, subject to raw material availability and delivery dates. The results show that the computational effort required by the heuristic increases with the planning horizon, and that it is substantially larger than for the CP model. However, the CP model requires more careful tuning, a topic that will be explored in further work. In any case, from a practical point of view, both methodologies proved feasible for planning factory operations, and none is particularly onerous for the desired two-week horizon. The actual penalties for constraint violations still have to be fine-tuned for the methods to be of use to the manufacturer.

We are aware of the existence of the Apparent Tardiness Cost with Setups and Ready times (ATCSR) composite dispatching rule introduced by [Pfund et al., 2008]. This rule could provide a better seed solution to the heuristic for the problem at hand; however, this has not been tested at the present stage and is left for future analyses.

ACKNOWLEDGMENT

The authors wish to acknowledge the students from Tecnológico de Monterrey that participated in the *Semester i* program in late 2016 for their important contribution to this paper.

REFERENCES

Allahverdi, A., J. Gupta, and T. Aldowaisan (1999). A review of scheduling research involving setup considerations. *Omega 27*(2), 219–239.

Blackstone, J., D. Philips, and G. Hogg (1982). A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *International Journal of Production Research 20*(1), 27–45.

Bockmayr, A. and J. Hooker (2005). *Handbook of discrete optimization – Constraint Programming*, pp. 559–600. Elsevier.

Brownlee, J. (2010). *Clever algorithms: Nature-inspired programming recipes.* LuLu. http://www.CleverAlgorithms.com.

Chen, J., M. Pfund, J. Fowler, D. Montgomery, and T. Callarman (2010). Robust scaling parameters for composite dispatching rules. *IIE Transactions 42*, 842–853.

Glover, F. and C. McMillan (1986). The general employee scheduling problem: an integration of MS and AI. *Computers and Operations Research 13*(5), 536–573.

Holthaus, O. and C. Rajendran (1997). Efficient dispatching rules for scheduling in a job shop. *International Journal of Production Economics 48*(1), 87–105.

Lamothe, J., F. Marmier, M. Dupuy, P. Gaborit, and L. Dupont (2012, June). Scheduling rules to minimize total tardiness in a parallel machine problem with setup and calendar constraints. *Computers and Operations Research 39*(6), 1236–1244.

Lawler, E., J. Lenstra, and A. Rinnoy-Kan (1982, July). *Deterministic and Stochastic Scheduling*, Volume 84, Chapter Recent developments in deterministic sequencing and scheduling: a survey, pp. 35–74. Dordrecht: Reidel.

Lee, Y., K. Bashkaran, and M. Pinedo (1997). A heuristic to minimize the total weighted tardiness with sequence-dependent setups. *IIE Transactions 29*(1), 45–52.

Lee, Y. and M. Pinedo (1997). Scheduling jobs on parallel machines with sequence dependent setup times. *European Journal of Operational Research 100*, 464–474.

Montgomery, D. (2005). *Design and analysis of experiments.* New York: Wiley.

Pfund, M., J. Fowler, A. Gadkari, and Y. Chen (2008). Scheduling jobs on parallel machines with setup times and ready times. *Computers and Industrial Engineering 54*, 764–782.

Pinedo, M. (2010). *Scheduling: Theory, algorithms and systems* (Fourth ed.). New York: Springer.