Portuguese Study Groups' Reports

Report on "Scheduling in a factory"

Problem presented by ForEver Procalçado at the
65th European Study Group with Industry
21^{st}–24^{th} April 2008
Centro de Matemática da Universidade do Porto
Portugal

March 24, 2009

Problem presented by:	Eng. Rui Russo (ForEver Procalçado)
Study group contributors:	J. Orestes Cerdeira, T. Charters, M. Cruz, R. Duarte, A. Guedes de Oliveira, C. Pinto, J. Rodrigues, P. Vasconcelos
Report prepared by:	J. Orestes Cerdeira[1], T. Charters[2], M. Cruz[3], P. Freitas[4], and P.B. Vasconcelos[5]

[1]Centro de Estudos Florestais *and* Dep. Mathematics, Instituto Superior de Agronomia, Technical University of Lisbon (TU Lisbon), e-mail: orestes@isa.utl.pt

[3]Dep. of Mechanical Engineering, Instituto Superior de Engenharia de Lisboa (IPL), Rua Conselheiro Emidio Navarro, 1, 1959-007 Lisboa, Portugal, *and* Centro de Física Teórica e Computacional, University of Lisbon, e-mail: tca@cii.fc.ul.pt

[2]Instituto Superior de Engenharia, Instituto Politécnico do Porto, e-mail: mbc@isep.ipp.pt

[4]Dep. of Mathematics, Faculdade de Motricidade Humana (TU Lisbon) *and* Group of Mathematical Physics of the University of Lisbon, Complexo Interdisciplinar, Av. Prof. Gama Pinto 2, P-1649-003 Lisboa, Portugal, e-mail: freitas@cii.fc.ul.pt

[5]Centro de Matemática da Universidade do Porto *and* Faculdade de Economia da Universidade do Porto, e-mail: pjv@fep.up.pt

Abstract

In order to carry out their orders of shoe soles, this company has a number of tasks $T_1, \ldots, T_n$ of different lengths to be assigned to groups of machines. Each group is operated by one worker (two in one case), and an operation cycle corresponds to injection, cooling, and removal of the sole. The time taken at each step varies from one order to another, and when starting a new task a machine needs to be tuned, which takes some extra time. Machines are working in parallel. At the moment the assignment is carried out empirically, and the problem proposed is to optimize the procedure.

1.1 Introduction

During the manufacturing of shoe soles at ForEver, each operator controls a machine which has to be set up everytime the production of a new model begins. This is followed by a tuning process, after which production follows a cycle consisting of the functioning of the machine proper, plus the operation by a worker who injects material into the mould, runs the machine and removes the produced sole.

This functioning is of a parallel nature, in the sense that machines operate independently one from another and, for the process considered here, are not part of a larger chain where they would need to wait for the input from another machine, for instance.

The running of this process includes thus the assignment and scheduling of the different sole models to different groups of machines and operators.

Simulação ESGI - Procalçado, SA
Ficheiro de dados entrada

Equipamentos

Máq	Nº de posições para moldes	Operadores atribuídos (inicial)	Nº Moldes/Operador (Inicial)
#1	2	1	2
#2	3	1	3
#3	3	1	3
#4	6	2	3

Encomendas a produzir

		Tamanhos													
Encomenda	Modelo Sola	36	37	38	39	40	41	42	43	44	45	Prazo entrega (dias úteis)	Tempo máquina (seg.)	Tempo manobra (seg.)	Tempo ciclo (seg.)
Enc. A	Lagos	230	320	320	120							2	120	20	140
Enc. B	Rios					1290	1765	1867	2341	1349	1143	5	180	90	270
Enc. C	Mares		50	75	90	120	150	279	289	200		3	180	25	205
Enc. D	Oceanos						3005	3505	3505	3005		7	120	35	155
Enc. E	Ribeiros	1045	1296	1296	1006							5	150	15	165
Enc. F	Regatos	5	5	5	5	5	5	5	5	5	5	1	140	30	170
Enc. G	Nascentes	10	25	25	20	10						2	130	25	155
Enc. H	Cascatas					707	927	1137	897	367	237	4	210	65	275
Enc. I	Pantanos	200	300	400	500	500	600	600	550	400		3	180	20	200

Todas as máquinas trabalham 22 Horas por dia e 5 dias por semana, excepto Máq #4 que só trabalha 15 horas por dia

Todos os modelos têm apenas 1 molde/tamanho, excepto Rios 43 (2 moldes).

Figure 1.1: A typical report of orders and available equipment.

The table given in Figure 1.1 shows an example of a list of orders together with the time taken to carry out each one, the required quantities and the corresponding deadlines. It also describes the number of machines and operators which are available to carry out such tasks. The problem is thus to decide on a sequence of task assignments satisfying certain objectives related to deadlines, optimization of worker's time, etc. More precisely, we want to partition the orders into batches ensuring that the above restrictions are met.

There are different time scales in this problem, the longest being the deadlines for each order. The remaining ones are shorter and are related to the making of a pair of shoe soles. These include the time that the machine takes to make the sole, the manipulation time done by a worker and a cycle time. All these time quantities are related to the making of a specific pair of shoe soles, and can be added up to make what we have called the processing time. There are also penalty times which measure the time required to change a mold in the same machine and the (longer) time

required to change the mold to a different one. These two are the key critical quantities that will, eventually, control the overall optimisation procedure that we are looking for.

We begin by giving a mathematical description of the problem which allows us to distinguish between the fundamental aspects and those which can be discarded in a first approach. We then provide a heuristic approach to solve this model and give an example which exemplifies how the method works. We then end with some conclusions and recommendations.

2.1 Mathematical formulation of the problem

For each job j (a pair of shoe soles from a specific model and of a given size), each machine m (a mold position), and each time instant t ($t = 0, 1, \ldots$), define a $0-1$ variable x_{jm}^t which is equal to 1 if and only if machine m starts to process job j at time t.

Let J denote the set of all jobs, p_j the processing time (which includes handling time) of job j, $O_k \subset J$ the set of jobs of order $k = 1, \ldots, |O|$, where $|O|$ is the number of different orders, and D_k the deadline of order k. Let M denote the set of machines and r_{ji} the time to adapt a machine to start processing job i immediately after finishing job j. Let also $J_s \subset J$ be the set of jobs which uses the same mold s (i.e., the set of shoe soles of the same specific model and of the same size), and n_s the number of molds of type $s \in S$.

The constraints can be stated as follows.

$$\sum_m \sum_t x_{jm}^t = 1,\ j \in J \tag{2.1}$$

$$\sum_m \sum_t (t + p_j) x_{jm}^t \leq D_k,\ j \in O_k,\ k = 1, \ldots, |O| \tag{2.2}$$

$$x_{jm}^t + x_{im}^{t'} \leq 1,\ m \in M, i, j \in J, t \leq t' \leq t + p_j + r_{ji} \tag{2.3}$$

$$\sum_{j \in J_s} \sum_{t'=t}^{t+p_j} \sum_m x_{jm}^{t'} \leq n_s, \quad s \in S, t = 0, 1, \ldots \tag{2.4}$$

$$x_{jm}^t \in \{0, 1\},\ j \in J, m \in M, t \in T. \tag{2.5}$$

Constraints (2.1) state that each job j will start to be processed at a unique time instant by a single machine. Constraints (2.2) ensure that all orders are finished on time. Conditions (2.3) ensure that there are no overlapping jobs assigned to the same machine. Inequalities (2.4) state that the number of overlapping jobs that use the same mold does not exceeds the number of molds. Finally, (2.5) are the 0–1 constraints on the variables.

The goal is to minimise the completion time of the last job. This can be

settled by

$$\sum_t (t + p_j) x_{jm}^t \leq F, \ j \in J, m \in M \tag{2.6}$$

$$\min F \tag{2.7}$$

Indeed, the left hand side in (2.6) is the time completion of job j on machine m. Thus, the variable F to be minimized is an upper bound on the time completion of the last job.

To find optimal solutions, even for moderate values of $|J|, |M|$ and $|T|$ (number of instants of time that should be considered), is a task that is likely to be impossible for an integer programming solver. Thus, a reasonable option is to conceive some heuristic procedure that produces *good* solutions.

3.1 A heuristic approach

A solution X of the problem can be viewed as a sequence of jobs on each machine, where each job occurs exactly once. Figure 3.1 represents a solution for a problem with 31 jobs and 5 machines.

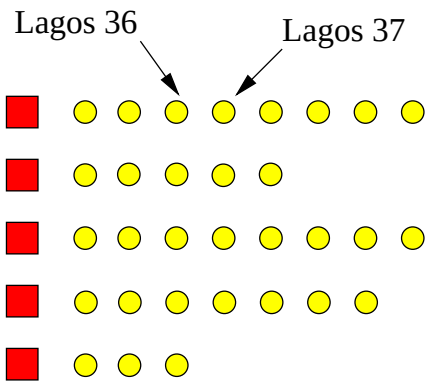

Figure 3.1: A representation of a solution with 31 jobs and 5 machines.

Define the *cost of solution* X, denoted by $c(X)$, to be the completion time of the last job of X. To minimise $c(X)$ is the goal of the model described in the previous section.

The solution X is not feasible if either

1. the completion time of the last job from some order k exceeds D_k (the deadline of order k), i.e., some inequality (2.2) is violated; or
2. the number of overlapping jobs that use the same mold, during some period of time, exceeds the number of molds, i.e., at least one of the inequalities (2.4) does not hold.

We suggest to deal with non feasibilities in a Lagrangian fashion (see for example [4]) incorporating the constraints in the objective function with associated multipliers. We illustrate this process with inequalities (2.2). Consider, for each solution X and $\boldsymbol{\lambda}= (\lambda_{jk}) \geq 0$,

$$w(X, \boldsymbol{\lambda}) = c(X) + \sum_{k=1}^{|O|} \sum_{j \in O_k} \lambda_{jk} (\sum_m \sum_t (t + p_j) x_{jm}^t - D_k). \tag{3.1}$$

Note that, if we set $\lambda_{jk} = 0$, for all jobs $j \in O_k$, whenever the completion time of the last job of order k does not exceed D_k, we have $w(X,\boldsymbol{\lambda}) \geq c(X)$, and $w(X,\boldsymbol{\lambda}) = c(X)$ if no inequality (2.2) is violated.

We propose to determine the set of orders $k \in O$ whose completions times exceed D_k (which is an easy task), and assign some positive value, say λ, to λ_{kj^*}, where j^* is the last job of order k. In this way, when minimising w we implicitly search for solutions which satisfy the constraints (2.2).

The above construction can be similarly developed to handle the constraints (2.4), using multipliers $\boldsymbol{\mu}$.

If X is an arbitrary solution, let T_k be the completion time of (the last job of) order k and let $\bar{O}$ be the set of orders k s.t. $T_k > D_k$. Let also $N_s = \max\{\sum_{j \in J_s} \sum_{t'=t}^{t+p_j} \sum_m x_{jm}^{t'}, \text{with } t \in T\}$ and $\bar{S}$ be the set of molds s s.t. $N_s > n_s$.

Once fixed positive values λ and μ, define the *weight of solution* X to be

$$w(X) = c(X) + \lambda \sum_{k \in \bar{O}} (T_k - D_k) + \mu \sum_{s \in \bar{S}} (N_s - n_s). \tag{3.2}$$

To minimise $w(X)$ a local search algorithm was designed. Local search heuristics iteratively move from one solution to another, by exchanging some of its components. The two basic ingredients are the rule describing the *neighbourhood* of every solution X, i.e., the set $\mathcal{N}(X)$ of solutions which can be directly obtained from X; and the criterium for choosing, for any solution X, an element in $\mathcal{N}(X)$.

In our problem, the neighbourhood $\mathcal{N}(X)$ could be defined as the set of all solutions that can be obtained by replacing the positions of two jobs in X.

Figure 3.2 represents two solutions in $\mathcal{N}(X)$, where X is a solution of an instance with 25 jobs and 5 machines

A local search heuristic which is particulary effective in finding near optimal solutions is *simulated annealing* (see, for example, [1]).

At iteration i of a simulated annealing algorithm, a solution X' which will be considered to replace the current solution X, is uniformly selected from $\mathcal{N}(X)$. If this improves the objective function w, then the current solution is replaced by X'. If $w(X') \geq w(X)$, it may still be replaced but only with probability $p = exp\left(\frac{w(X)-w(X')}{T_i}\right)$, where $T_i > 0$, the solution

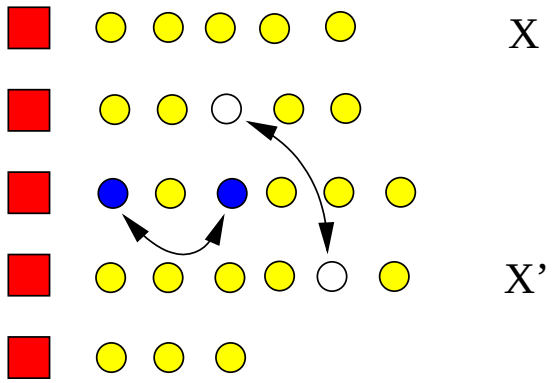

Figure 3.2: A solution X of an instance with 25 jobs and 5 machines and two of its neighbours.

X remaining the current solution with probability $1 - p$. The theoretical convergence of the algorithm to an optimal solution, when $\{T_i\}$ converges to zero, is ensured given some quite general conditions regarding the structure of the neighbourhood, the selection rule, and the rate of convergence of $\{T_i\}$.

A description of the algorithm for the problem in consideration follows.

Choose $\lambda, \mu > 0$.

let X be an initial solution, N the number of iterations and $i = 0$.

1. choose $T > 0$ (the initial temperature);
2. define $w(X)$ according to (3.2);
3. let X' be uniformly selected from $\mathcal{N}(X)$;
4. define $w(X')$ according to (3.2);
5. if $w(X') \leq w(X)$ let $X := X'$ (accept neighbour X');

   else
   draw R uniformly from $[0, 1[$;
   if $exp\left(\frac{w(X)-w(X')}{T_i}\right) > R$ let $X := X'$ (accept neighbour X');
6. let $i := i + 1$;
7. if $i \leq N$ stop (end program);

   else
   let $i := i + 1$;
   update T (decrease the temperature)
   go to 3;

4.1 A simple example

In this section we show the results of a basic application of the method described. For simplicity we use only some of the restrictions, but implementing the more general version with the full set of restrictions, although more involved, is not more difficult from a conceptual point of view. Basically we considered the case where we want to distribute a set of orders among a set of machines, taking into account the ending time and restrictions which include both the time to change from one mould to another and between machines. The implementation was done using Matlab.

As an example take a set of orders of the following sizes:

$$2 \quad 5 \quad 10 \quad 10 \quad 12 \quad 20 \quad 22 \quad 30 \quad 30 \quad 30 \quad 40 \quad 50 \tag{4.1}$$

totalling 261 jobs, which we want to distribute among three machines. The numbers were chosen just for illustration and the program handles any other distributions – this should be seen as just an example to illustrate the results and were we kept things simple.

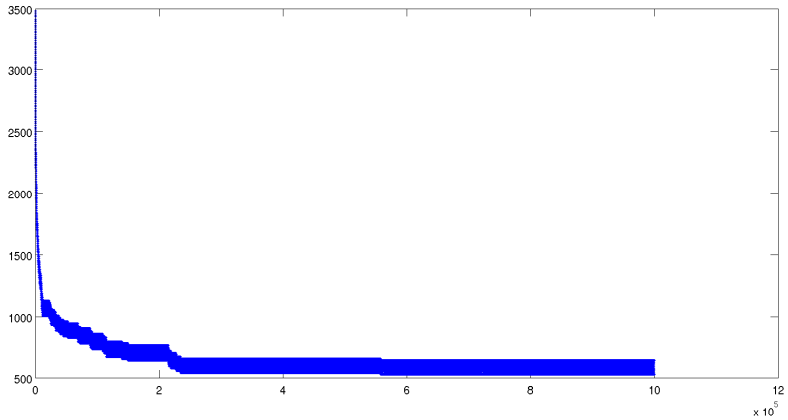

Figure 4.1: Evolution of the total time taken for the example described by the set in (4.1).

In Figure 4.1 we depicted the evolution of the cost function, while Figure 4.2 gives a schematic presentation of the jobs through the three machines. This last graph should be read from left to right, and each of the symbols represents one of the orders. The first two red diamonds at the start of machine 1 correspond to the (only) order of this size, while the order of size 20, represented here by black-filled squares, has been partitioned into three blocks of size 4, 6 and 10 which have been assigned to machines 3, 1 and 3, respectively.

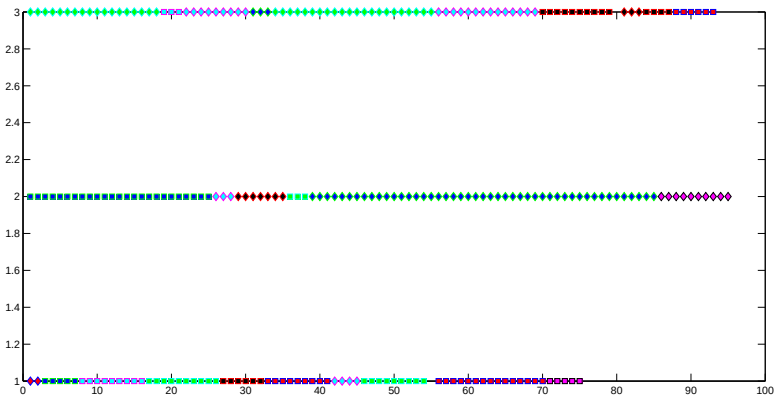

Figure 4.2: One possible partition solution for the example described by the set in (4.1).

We stress that a partition obtained by this method such as the one above, is not necessarilly optimal. For the example of the order of size 20 mentioned above, it would be possible, for instance, to lump the two blocks being processed by machine number 3 into one. However, this would not necessarilly mean a big gain regarding the solution already obtained. More important than this, the solution provided satisfies the given constraints and has already been through an optimization procedure which ensures a certain quality standard. Furthermore, this approach is very flexible and by changing the constraints given in the mathematical formulation of the problem (see (2.5)), it is possible to modify the goals that are to be achieved with a minimum conceptual effort.

5.1 Conclusions and recommendations

Scheduling problems are common in industrial environments, varying from simple processes to highly non trivial tasks assignments. In real situations deriving from concrete industrial setups, this will usually give rise to hard computational problems. This means that, in general, it will not be possible to achieve the optimal solution due to time constraints, but one will have to use a heuristic algorithm instead. In the case of the proposed problem, our analyis lead us to a simulated annealing algorithm which is quite flexible and allows us also to deal with variations of the problem considered. This may be achieved by changing the constraints in an appropriate way.

The proposed approach is based on a Lagragian manipulation of a $0-1$ model, which means that tuning operations will have to be developed in order to obtain near-optimal solutions. The main concerns for an imple-

mentation should be:

1. The choice of the multipliers λ and μ. Larger values of these parameters will put a stronger emphasys on the admissibility conditions. A careful equilibrium must be achieved in order to balance feasibility with the capacity to distinguish between the quality of different solutions.

2. The definition of the initial parameter T (temperature) for the simulating annealing algorithm and how to actualize this parameter during the process. Several different approaches are possible here (see for example [2].

Even in the case where a heuristic algorithm is being used, one might have to take into consideration the fact that the number of jobs considered might be too large to allow us to produce a solution in real time. For the case at hand, where the example given in the Introduction had around 30 000 jobs, there may already be some difficulties in dealing with such large numbers. This problem may be solved by making a preliminary choice of what the unit job should be which is different from the one made here. This will depend on the size of the order, and different choices might have to be made for different orders. From the table in Figure 1.1 we see that, for instance, order F should have as a minimum job size five units, as not only it is highly unlikely that these should be partitioned but also should this be the case, the gain is likely to be very small. Other orders should have different basic job units, depending on their overall size, and this might be another parameter that will also need some tuning: decide upon the minimal job unit depending on the total order size. However, once this learning process is over, the actual scheduling should run quite smoothly.

Bibliography

[1] E.H.L. Aarts, J.H.M. Korst and P.J.M. van Laarhoven, "Simulated annealing", in *Local Search in Combinatorial Optimization* (E. Aarts and J.K. Lenstra, *eds.*), John Wiley & Sons (1997), 91-120.

[2] P.J.M. van Laarhoven and E.H.L. Aarts, Simulated Annealing: Theory and Applications, Springer, 1987.

[3] C.L. Liu, and J.W. Layland,Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment, Journal of the ACM , (1973)

[4] G.L. Nemhauser and L.A. Wolsey, Integer and combinatorial optimization, Wiley-Interscience, New York, NY, 1988