Flow shop operator scheduling through constraint satisfaction and constraint optimisation techniques

Gilles Neubert*
Laboratoire LIESP – School of Management,
51-53 cours Fauriel, 42000 Saint-Etienne, France
E-mail: gilles_neubert@esc-saint-etienne.fr
*Corresponding author

Matteo M. Savino
Department of Engineering - University of Sannio,
Piazza Roma, 21 – 82100 Benevento, Italy
E-mail: matteo.savino@unisannio.it

Abstract: Workers scheduling in not highly automated production lines is an important task, especially when in a production line the number of operators is less than the number of workstations. Finding an optimal distribution plan can increase the line throughput, managing the workforce and the workload in a better way. This work focuses on the operator-scheduling problem for an electromechanical assembly line. Workforce distribution on the workstations has been made with a centralised scheduling based on a mathematical model which, through constraint optimisation principles, is able to find the optimal distribution of workforce optimising fundamental parameters, such as man-hours, throughput, makespan and work in process.

Keywords: operator scheduling; constraint optimisation problem; COP; constraint satisfaction problem; CSP; workforce distribution.


Biographical notes: Gilles Neubert is Head of Postgraduate programs at Saint-Etienne School of Management (SESOM). He is a member of the Supply Chain team of LIESP laboratory (Lyon-France). He received his PhD in 1997 in Industrial Engineering from INSA National Institute of Applied Sciences of Lyon. His main research topics concern supply chain performance, processes engineering and modelling. Before joining SESOM, he was an Associate Professor at University Lumiere of Lyon. He was the Founder of the Production Management Department of the Technology Institute of LYON2 University and the Head of this department from 1998 to 2002.

Matteo Mario Savino received his PhD in Industrial Operations in 1998 at University of Naples, Department of Industrial Engineering. From 2002, he is Professor of Industrial Operations and Management at Samnium University, Benevento, Italy, for the Master Courses in Automation Engineering and Energetic Engineering. He has directed two masters on ERP systems and several international projects. Actually, he is a Coordinator for his University of the eLink Erasmus Mundus project, involving 15 countries, and of a
1 Introduction

An assembly line is a series of workstations at which people or machines add to or assemble parts for a product. This is an appropriate choice when the product demand is high enough and investment is low enough, to specialise a flow shop to the production of a standard unique product. This process is often referred to as mass production. Semi-finished units move through consecutive stations where workers generally perform the same operations for each production run in a standard uninterrupted flow. For these reasons, an assembly line has the advantages of part standardisation and rationalisation of work.

The set of tasks to be performed in the same station is called an ‘operation’ and an assembly line is designed by determining the number of workstations and the sequences of operations. This is well-known, as the assembly line balancing problem (ALBP) consisting in assigning operations to workstations along the line, in such a way that the assignment can be optimal.

Once the assembly line is designed, its production rate could be limited by the number of active workstation within a time slot. Especially if the number of workers is less than the number of workstations, the assembly line planning consists in finding the optimal assignment of the workers to the workstations during the day. In that case, if all workstations are not running at the same time, this process needs buffer between workstations. Another constraint is that operators need to work on different workstations during a day and for this reason need to be multi-skilled. In this condition, there are alternative ways of organising the flow line activities. One possible solution is to let the workers autonomous in their organisation; in this case, global optimality results from the collaboration and synergy of the operators. Another solution is to define the planning of each operator in the line. In this case, workers’ shifts optimisation is important to increase the line throughput, while a correct production planning can improve the production level and facilitate workers’ management. Furthermore, with an optimal distribution of the human resources around the lines, it is possible to avoid additional costs (the firms can distribute workforce in a better way and the workstations are inactive when it is convenient).

The present work proposes a model to generate an optimal operator’s planning of an assembly line with fewer workers than workstations. The optimal solution to this operator-scheduling problem is dedicated to the improvement of three production performance parameters: man-hours, makespan and throughput. The operator-scheduling problem attempts to assign operators to time slots of equipment utilisation. As it is different from the classical job scheduling problem, well-known algorithms using job characteristic cannot be used. For this reason, this paper will present an original approach using constraint satisfaction problem (CSP) technique to obtain an optimal planning of the operators.

In the first part of the paper, a literature review on the operator scheduling problem and CSP is given, while in the second part a case study is discussed. Based on an
assembly line with fewer workers than workstations, the constraint optimisation problem (COP) is applied to find an optimal solution to the operators’ planning.

2 Operator-scheduling problem

Among the decision problems, which arise in managing the described production systems, the ALBPs are important tasks in medium-term production planning. The ALBP has to be solved when an assembly line has to be configured or redesigned.

The problem of balancing an assembly line is a classical industrial engineering problem.

It is possible to characterise an ALBP in the following way (Abbas and Tsang, 2001):

“A set of $n$ tasks that must be completed on each item. The time required to complete task $i$ is a known constant $t_i$. The goal is to organise the tasks into groups, with each group of tasks being performed at a single workstation. In most cases, the amount of time allotted to each workstation (this is the cycle time) is determined in advance, based on the desired production rate of the assembly line.”

With this definition, the main problems that already remain to be answered are mainly related to:

- Given a certain level of cross training, how should these workers be allocated to the workstation?
- In addition, when a worker has finished his task at a station, how should he select the next station?

These questions, which are the essential of the operator-scheduling problem, are very important to the managerial point of view and the existent literature tries to give an answer to them. A number of authors have indicated that the increase of workers’ flexibility seems to improve systems performance (Daniels et al., 2004): ‘In production-line environments such as these, partial (labour) resource flexibility is a particularly important issue that is determined by the extent to which workers are cross-trained to perform a subset of the tasks occurring within the line’. Hopp et al. (2004) have demonstrated that chaining can be useful in an assembly line environment, i.e., task time variation and worker absenteeism.

So, there are a variety of factors that contribute to increase the complexity of the problem, one of these (fundamental when the human presence to the workstations is required) is to solve the above mentioned operator-scheduling problem.

The operator-scheduling problem attempts to assign operators to time slots of equipment utilisation while satisfying the requirement for the number of operators at a specific time slot, various regulations for workforce utilisation and the requirements of operators for assignments.

In this context, the problem arises when the number of operators is inferior to the number of stations. It consists in dividing the work time in time slots (often one hour) and to assign each available operator to a specific station during each time slot. At the end of the time slot, operators have to move from one station to another, with respect to the various constraints of the production cycle and so on, finally performing every operation on every workstation.
In this work, the operator-scheduling problem is different from the previous research on the roster-scheduling problem. Generally, in the previous studies the workforce requirement was expressed as the total number of workers required during a period (Hwang et al., 2004). On the contrary, in this study, the requirement can be expressed as an operating schedule for multiple pieces of equipment, which makes the problem more complicated.

So, it is possible to say that the operator-scheduling goal, in this case, is to determine:

- the workstation to which each operator will be assigned during each working time slot
- the work schedule of each operator during a planning horizon.

Due to the problem difficulty, it is impossible to obtain a good solution with classical scheduling algorithms, in fact, methods like FIFO, LIFO, EDD, SJF and similar are based on the jobs and not on the labour. As well-known, these algorithms schedule workpieces, because they are based on specific characteristic (arrival time, due date, cycle time, priority, penalty costs, etc.) that can determine, which job is better to schedule first of all, so it is not correct to apply these strategies to human resources because they do not have the same (or similar) characteristics, therefore, it is necessary to adopt other types of strategies. In this work, the problem will be approached using CSP to obtain a ‘centralised’ scheduling.

2.1 CSP and COP

Many problems can be cast as CSPs: diagnosis and temporal reasoning (Allen, 1984; Tsang, 1987), design, graph problems (Barber and Salido, 2006), timetabling (Abbas and Tsang, 2001), flight crews scheduling (Graves et al., 1993). Another field in which CSPs are used is scheduling, for which the literature offers extensive references: Cheng and Smith (1997) apply CSP to a deadline scheduling ‘to provide a basis for high-performance approximate solution procedures in optimisation context’, another example of CSP applied to scheduling is given by Cheng and Smith (1997). Their paper reports an ongoing project of applying CSP algorithms to solving job shop scheduling problems for the characterisation of feasible schedules. CSP has also been applied to maintenance scheduling; Daniel and Dechter, (1999) show the solution of a problem of preventive maintenance scheduling for power generating units within a power plant.

Another classic example of CSP problem is the ‘bucket brigade’ problem: Bartholdi and Eisenstein (1995) introduced this term to indicate a system with multi-skilled flexible workers that are allowed to search for their own optimal place along a production line. To describe this problem, they observed and described the assembly line named ‘Toyota Sewn Products Management System’, or TSS, a kind of line that is used to realise many types of sewn products as, for example, apparel, furniture, shoes, hand bags, suitcases and fishing nets.

In this case, instead of fixed work allocation, a set of policies indicates to each worker what to do next. Their experiments have proved that if the workers are sequenced from slowest to fastest, independently of the stations at which they begin, a stable partition of work will spontaneously emerge. Furthermore, the production rate will converge to a value that, for these types of production lines, is the maximum possible.
The problem of workforce scheduling with more workstation than workers has been also discussed by other authors as Felan et al. (1993) who consider two identical machines for a department in a job shop and Cheng et al. (1999) considers a problem of scheduling of one operator and two machine in a flow shop, solving it with an heuristic.

A CSP involves assigning values to variables, satisfying a set of constraints, which may take an arbitrary form, such as databases or functions (Tsang, 2002).

A CSP can be defined (Beck et al., 1998; Borret, 1998; Tsang et al., 1999) by a triple \((Z, D, C)\)

Where:

- \(Z\) is a set of variables \(Z = \{z_1, \ldots, z_n\}\)
- \(D\) is a set of domain \(D = \{D_1, \ldots, D_n\}\)
- \(C\) is a set of constraints \(C = \{R_1, \ldots, R_m\}\)

The set of domain \(D\) is a function that maps each variables in \(Z\) to a domain (set of values of arbitrary type): in other words \(D_i\) is a finite set of values that can be assigned to variable \(z_i\). A constraint \(R\) restricts the value that the variable can simultaneously take.

In CSP, the task is to find a value for all variables in \(Z\), such that all constraints in \(C\) are satisfied.

With respect to assembly lines, it is possible to define the constraints as everything that limit management to obtain the higher performance level, where performance is related to organisation objective (Mukhopadhyay and Panda, 2007). So, the constraints establish the upper bound of system performance (Gunasekaran et al., 2003).

CSP are combinatorial in nature and solving this kind of problems is \(Np\)-Hard, therefore, general algorithms are likely to require exponential time in the worst case. Thus, in practice, it may be sufficient to find a solution at a reasonable computational time that satisfies most of the constraints. Most of the approaches have been used to solve CSP: integer programming techniques, as branch and bound (Lawler and Wood, 1966), that find an exact solution; local search and heuristics algorithms (Baker and Burns, 1979) that provide an approximate solution.

If instead there is an objective function to minimise or maximise, we front the so-called COP.

A COP is a particular class of CSPs. In this type of problem, some constraints can be cost functions, which indicate certain preferences. The constraints of a CSP are called hard constraints (i.e., that have to be necessarily satisfied), while the cost functions are called soft constraints (i.e., they only express a preference of some solutions over other ones).

A COP is defined as:

\[
(Z, D, C, f)
\]

where:

- \(Z\) is a set of variables \(Z = \{z_1, \ldots, z_n\}\)
- \(D\) is a set of domain \(D = \{D_1, \ldots, D_n\}\)
- \(C\) is a set of constraints \(C = \{R_1, \ldots, R_m\}\)
- \(f\) is the objective function.
The new task is now to find a value for all variables in $Z$, such that all constraints in $C$ are satisfied and $f$ is maximised (or minimised). An ‘optimal solution’ consists of an assignment of values to the $z_i$ such that all constraints in $C$ are satisfied (i.e., the solution is valid) and the assignment yields an optimal value for the objective function $f$.

Also, this problem is NP-complete and can be solved by the techniques presented above for CSPs.

In general, selecting a scheduled rotation policy presents a constrained optimisation problem, in which workers are resources to be scheduled according to an objective function.

Several studies have focused on the application of COP for labour scheduling in different industrial contexts such as airlines, railways and buses, healthcare and call centre (Beasley and Cao, 1998; Ferland and Taillefer, 1992; Graves et al., 1993; Berman et al., 1997; Campbell, 1999; Abril et al., 2007). Easton and Mansour (1998) propose a genetic algorithm to solve the problem of number of employees and their work schedule to minimise the labour expense and expected opportunity costs. Another example of COP application is given by Burke et al. (2004); they describe the nurse scheduling problem (NSP), that involves the construction of duty rosters for nursing staff and assigns the nurses to shifts per day, taking into account both hard and soft constraints. The objective is to maximise the preferences of the nurses and to minimise the total penalty costs from violations of the soft constraints. To solve this problem, Burke mentions simulated annealing, tabu search and genetic algorithms as popular meta-heuristics.

The industrial case described below is a COP problem: a CSP technique has been used, aiming at modelling the problem and assigning a set of values to the variables that satisfy all the constraint while maximising the daily production.

3 The assembly line

As mentioned before, this work addresses a problem related to an assembly line (Figure 1) in which there are more workstations than workers and each workstation, at a certain time slot, has to be attended by a worker. The workers are dependent on each other, because the output of a worker is the input to another worker. In addition, because there are more workstations than workers deployment, rules are necessary to guide each worker to search the next workstation when he has finished his work on the previous one. The investment in cross-training and effective labour coordination mechanisms can yield dividends that include not only greater operational efficiency, improved job satisfaction and higher quality, but also increased organisational flexibility to deal with unforeseen change.

This production system can be classified as a flow shop scheme (Nahmias, 1998). In this case, each job is characterised by a technological cycle that needs more different machines. The sequence of operations is the same for every job. Especially this assembly line can be classified as a pure flow shop because every job needs one operation at each workstation (Daniels et al., 2004).

The operator -cheduling problem will be front for the first six workstations (named as WS1, WS2, WS3, WS4, WS5 and WS6 in Figure 1) in which the most part of assembly work is made.
The constraints related to the workstations are the following:

1. **Ws**₁, **Ws**₂ and **Ws**₃ have to work in the same time slots
2. **Ws**₁, **Ws**₂ and **Ws**₃ need two workers
3. as cycle time of the third workstation is very short, the same operator will be on **Ws**₁ and **Ws**₃ (these constraints are due to the design of the line and cannot be modified).
After each station there is a buffer so the data associated to buffers capacity will be also considered, together with workstations cycle time (these data are shown in Table 1 and Table 2).

**Table 1** Cycle time for one part, measured on the line

<table>
<thead>
<tr>
<th>Time 1 (seconds)</th>
<th>Time 2 (seconds)</th>
<th>Time 3 (seconds)</th>
<th>Average (seconds)</th>
<th>Average (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WS_1</td>
<td>50.70</td>
<td>41.70</td>
<td>47.40</td>
<td>46.60</td>
</tr>
<tr>
<td>WS_2</td>
<td>107.30</td>
<td>88.80</td>
<td>94.60</td>
<td>96.90</td>
</tr>
<tr>
<td>WS_3</td>
<td>16.20</td>
<td>15.10</td>
<td>14.60</td>
<td>15.30</td>
</tr>
<tr>
<td>WS_4</td>
<td>111.80</td>
<td>96.70</td>
<td>102.02</td>
<td>103.57</td>
</tr>
<tr>
<td>WS_5</td>
<td>85.40</td>
<td>62.94</td>
<td>103.80</td>
<td>84.05</td>
</tr>
<tr>
<td>WS_6</td>
<td>124.00</td>
<td>94.80</td>
<td>109.30</td>
<td>109.37</td>
</tr>
</tbody>
</table>

**Table 2** Buffers capacity

<table>
<thead>
<tr>
<th>Buffers</th>
<th>b_0</th>
<th>b_1</th>
<th>b_2</th>
<th>b_3</th>
<th>b_4</th>
<th>b_5</th>
<th>b_6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity</td>
<td>inf.</td>
<td>20</td>
<td>0</td>
<td>20</td>
<td>6</td>
<td>20</td>
<td>inf.</td>
</tr>
</tbody>
</table>

The line that is object of optimisation is characterised by:

- three operators
- six workstations
- seven buffers (the index will go from 0 to 6)
- each roster is divided in eight work hours.

A conceptual schema is provided in Figure 2

**Figure 2** Conceptual schema of the six workstations (see online version for colours)

3.1 *The mathematical model*

In this section, the mathematical model adopted to compute the centralised scheduling will be discussed.

The idea on which the model is based is to represent the quantity or the level of each activity by a variable. Using the variables, it is possible to write a set of relations, which mathematically could explain all the limits related to planning or scheduling problem. The problem solution to every constraint will be the admissible plan or schedule.
It is known that it is difficult to write a model associated to a complex operation just using constraints (Abbas and Tsang, 2001). Too few constraints can give mediocre results, while too many constraints can give better results but also, in the worst case, no solutions (it could be impossible to satisfy all constraints). We have assessed this problem by adding an objective function related to the ‘decision variables’, determining in this way the best solution COP by the maximisation or minimisation of an objective function. In our case, we have considered the following two parameters:

- the daily productive capacity
- the number of work hours for each operator.

With this goal, the mathematical model of the line is written with the following notations:

- \( op = \{1, \ldots, op_{num}\} \) the number of available operators
- \( w = \{1, \ldots, w_{num}\} \) the number of workstations
- \( s = \{0, \ldots, s_{buffers}\} \) the buffer quantity
- \( k = \{1, \ldots, k_{shifts}\} \) the workday length (in hours).

As decision variables we have:

\[
x_{w}^{op}(k) \quad \text{operator’s state; with this variable it is possible to describe if an operator is working near a workstation or not.} \quad x_{w}^{op}(k) = \{1 \text{ if the worker } op \text{ is working on workstation } w \text{ at time } k, 0 \text{ if the worker is idle}\}
\]

\[
p_w(k) \quad \text{pieces worked on the workstation } w \text{ at time } k
\]

\[
b_{s_{buffers}}(k_{shifts}) \quad \text{number of pieces present in the last buffer; with this variable it is possible to describe the number of jobs that have finished their operations and are ready for other steps (or to be sold).}
\]

With these variables, the objective function created for these goals is the following (3.2):

\[
\max \quad \lambda \cdot b_{s_{buffers}}(k_{shifts}) - \sum_{k=1}^{k_{shifts}} \left( \sum_{w} x_{w}^{op}(k) \cdot (\gamma - \varphi \cdot k) \sum_{w} p_w(k) \right)
\]

where:

- \( \lambda \) is a specific weight to weigh up the throughput
- \( \gamma \) and \( \varphi \) are constants used to give precedence to the first workday hours. As the function has to be maximised, in this way we have:
  \[
  (\gamma - \varphi \cdot 1) > (\gamma - \varphi \cdot 2) > \ldots > (\gamma - \varphi \cdot k_{shifts})
  \]

To write the function in a right way, the three constants have to satisfy the following:

\[
\lambda, \gamma, \varphi > 0 \quad \gamma > \varphi \cdot k_{shifts} \quad \lambda >> \gamma
\]
The third relation indicates that in the objective function the first term must have always more weight than the last one.

The first term of the (3.2) tries to make more jobs as possible at the end of the line (finished jobs) as the function will try to maximise the variable $b_{\text{buffers}}(k_{\text{shifts}})$ that represents the total production.

The second term of the (3.2) is associated to operators’ scheduling; it reduces (the minus in the function) the shifts’ number, so the workforce will be distributed in an optimal way with respect to its total work hours. In the following section, we will show that this term is used also to implement one of the constraints. The last term of the (3.2) has been introduced to avoid that the work is developed at the end of the workday; it is more realistic to see labours doing their work at the beginning than at the end of the day (in this way it is possible to spend the remaining time to do other things if necessary).

### 3.2 Problem constraints

After objective function definition, the next step is to implement all the assembly line constraints. As one of the goals is to maximise the production volume, it is necessary to implement the buffer dynamics as it is important to keep trace of how the jobs go ahead in the line.

The first constraint is related to the initial condition, for each buffer we have created linear equation to fix the pieces number contained at time 0:

$$b_s(0) = b_s^0 \quad \forall s \quad (3.3)$$

Where $b_s^0$ is a constant vector (of length $s$) with all the initial condition for each buffer.

Now, it is necessary to model how the jobs advance between the workstations, for the first buffer:

$$b_s(k) = b_s(k-1) - p_1(k) \quad (3.4)$$

Obviously, the worked quantity at time $k$ by first workstation – $W_{s1}$ – will be subtracted from the buffer and, with another constraint; it will be added to the following buffer (as workstation output). In the other cases (for the other buffers), the dynamic will be governed by the following equation:

$$b_s(k) = b_s(k-1) - p_{s+1}(k) + p_s(k) \quad (3.5)$$

The (3.5) gives, for each buffer, the worked pieces by the previous workstation (they will be added) and by the following one (they will be subtracted). As for the first buffer, there is not the ‘previous’ workstation, for the last buffer there is not the ‘following’ workstation, so:

$$b_{\text{num}}(k) = b_{\text{num}}(k-1) + p_{\text{num}}(k) \quad (3.6)$$

The last constraint associated to buffers is the upper bound that is specified as:

$$b_s(k) \leq b_s^{ub} \quad (3.7)$$

At each time slot, each buffer must remain under its limit.
After the dynamic constraints, it is necessary to implement all limitations related to workers.

The first thing about workers is that they can work just on one workstation at the same time (except for the first and third workstation, Ws1 and Ws3, as described by the third constraint written in Section 3). To implement this constraint, it is sufficient to sum all the state variables related to single operator and to put an upper bound to this sum equal to one; as said by (3.1) the variable value is 1 if the worker is on that workstation so a sum equal to 2 (or more) is wrong because it means that the operator is working on two (or more) workstation and this situation goes against hypothesis made before (Section 3), therefore, the constraint will be written in the following way:

$$\sum_{w \in \mathcal{W}} x_{w}^{op}(k) \leq 1$$

(3.8)

If the sum is greater than 1, it means that the worker is contemporarily present on more workstations and this is impossible.

In a similar way, one has to avoid that for one workstation there are 2 (or more) operators, so it is sufficient to put an upper bound (equal to 1) to each operators summation related to each workstation:

$$\sum_{w} x_{w}^{op}(k) \leq 1$$

(3.9)

If the sum is greater than 1 it means that two workers are contemporarily present on the same workstation and this is impossible.

Now, to satisfy the constraints 1, 2 and 3 in Section 3, it is necessary to write other two equations. The first one insures that workstation Ws1 can work only if also the workstation Ws2 is ‘on’ (in this way the contemporary presence, or absence, of two workers on workstations is ensured), so:

$$\sum_{op} x_{1}^{op}(k) = \sum_{op} x_{2}^{op}(k)$$

(3.10)

The second one have to satisfy the condition that workstations Ws1 and Ws3 are assigned to the same operator, thus the associated equation is:

$$x_{1}^{op}(k) = x_{3}^{op}(k) \quad \forall op$$

(3.11)

Consequently, for each operator the state on the first and third machine is the same.

Another constraint related to pieces dynamic, that derives from the equations (3.10) and (3.11), i.e., the total time spent to work the pieces sum in workstation Ws1 and Ws3 must be less than 60 minutes (as time slot duration is one hour) since the worker is the same:

$$t_{1}p_{1}(k) + t_{3}p_{3}(k) \leq 60$$

(3.12)

It remains the last constraint, probably the most important.

As the main goal of this work is to schedule the available operators in an optimal way to increase the total production, the crucial point is to associate the workforce and the production on each workstation. To be able to run, workstations need to have one
operator near them, otherwise they cannot produce. This is a particular problem, as we should model the following:

\[ p_w(k) > 0 \Rightarrow x^{op}_w(k) = 1 \]
\[ p_w(k) = 0 \Rightarrow x^{op}_w(k) = 0 \]  (3.13)

To front this problem, we have assumed that if there is a known upper bound \( \alpha \) to the production variable \( p_w(k) \) it is possible to write:

\[ p_w(k) - \alpha \cdot x^{op}_w(k) \leq 0 \]  (3.14)

Obviously, any operator can work on the machine \( w \) so it is necessary to extend the operator state variable to all workers; we have expressed it by summing the (3.14) on every operator, i.e.,

\[ p_w(k) - \alpha \cdot \sum_{op} x^{op}_w(k) \leq 0 \]  (3.15)

The constraint expressed by the (3.15) forces the sum to be equal to 1 if the production is more than 0. In this way, it is impossible that pieces can be worked without operator. In fact, when the production is more than 0, if the sum is null, the total is positive and the constraint is not satisfied.

The second part is not yet complete because it is also necessary to express that when there is no production no worker must be on the workstation. This fact is modelled whit a little help from the objective function. As in the objective function, there is a ‘negative’ term related to workers’ job (to model their work day hours) and as the function will be maximised, the solution automatically will keep the operator state variable to 0 if there is no production on the associated workstation. Of course in the objective function the worked pieces have more weight than workers’ hours so no problem occurs if there is production and the solution will try to make more pieces as possible because that is the only way to maximise the function (even to put the pieces into the last buffer it is necessary to produce them). Thanks to the other constraints and to the objective function it is easy to see that also the ‘vice versa’ is verified, so:

\[ p_w(k) > 0 \Leftrightarrow x^{op}_w(k) = 1 \]  (3.16)
and

\[ p_w(k) = 0 \Leftrightarrow x^{op}_w(k) = 0 \]

Finally, we have to specify the ‘production upper bound’ \( \alpha \), that has been set using the upper bound related to the worked pieces per hour (3.17).

\[ p_w(k) - \frac{60}{t_w} \sum_{op} x^{op}_w(k) \leq 0 \]  (3.17)

where \( t_w \) is the cycle time and so the ratio \( t_w/60 \) represents the maximum throughput of the workstation. This constraint guarantees that the workstation can work only if there is a worker and it sets (with the ratio) an upper bound to worked pieces per hour. Moreover,
Flow shop operator scheduling

561

together with the second term of the objective function, this constraint avoids that a worker is matched with a workstation and he does not produce pieces.

Now the problem description is complete and it is possible to write the entire COP (3.18), solving it by a professional solver. Analysing the model, it is clear that this problem is included among the integer programming family as the decision variables must have an integer value.

$$\max \kappa \cdot b_{\text{shift}}\left(k_{\text{shift}}\right) - \sum_{k=1}^{k_{\text{shift}}} \left(\sum_{w} x_{w}^o (k) - (y - \phi \cdot k) \sum_{w} p_{w} (k)\right)$$

subject to:

$$b_{\text{h}} (0) = b_{\text{h}}^0$$
$$b_{\text{h}} (k) = b_{\text{h}} (k-1) - p_{1} (k)$$
$$b_{w} (k) = b_{w} (k-1) - p_{w+1} (k) + p_{w} (k)$$
$$b_{w_{\text{max}}} (k) = b_{w_{\text{max}}} (k-1) + p_{w_{\text{max}}} (k)$$
$$b_{\text{h}} (k) \leq b_{\text{h}}^\text{ab}$$
$$\sum_{w \in \{3\}} x_{w}^o (k) \leq 1$$
$$\sum_{o} x_{o}^o (k) \leq 1$$
$$\sum_{o} x_{o}^o (k) = \sum_{o} x_{o}^o (k)$$
$$x_{1}^o (k) = x_{2}^o (k)$$
$$\ell_{1} p_{1} (k) + \ell_{3} p_{3} (k) \leq 60$$
$$p_{w} (k) - \frac{60}{\ell_{w}} \sum_{o} x_{o}^o (k) \leq 0$$
$$x_{w}^o (k) = \{0, 1\} \quad \forall o, w, k$$

$$b_{\text{h}} (k) \in \square$$
$$p_{w} (k) \in \square$$

3.3 Comparative examples

In this section, we present some examples related to the application of the described algorithm. For the same example, we provide two solutions; the first one will be obtained by the presented algorithm using ILOG as a solver, the second one will be made without it, but simulating a human, decision taken by the production manager of the case study. The solution of the model has been made through ILOG Optimisation Programming Language (OPL), this is an integrated development environment (IDE) for mathematical programming and combinatorial optimisation applications. It is the graphical user interface (GUI) for the OPL modelling language (ILOG, 2006).

The line starts from the initial conditions in Table 3.
Table 3  Initial conditions

<table>
<thead>
<tr>
<th>Buffers</th>
<th>b0</th>
<th>b1</th>
<th>b2</th>
<th>b3</th>
<th>b4</th>
<th>b5</th>
<th>b6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantity</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>3</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

Case A

An apparently good solution to the scheduling problem, starting from the described initial condition, is shown in Table 4.

Table 4  Case A

<table>
<thead>
<tr>
<th>Timeslot 1</th>
<th>Timeslot 2</th>
<th>Timeslot 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ws1</td>
<td>Op1</td>
<td></td>
</tr>
<tr>
<td>Ws2</td>
<td>Op2</td>
<td></td>
</tr>
<tr>
<td>Ws3</td>
<td>Op1</td>
<td></td>
</tr>
<tr>
<td>Ws4</td>
<td>Op1</td>
<td>Op3</td>
</tr>
<tr>
<td>Ws5</td>
<td>Op3</td>
<td></td>
</tr>
<tr>
<td>Ws6</td>
<td>Op2</td>
<td></td>
</tr>
</tbody>
</table>

This plan seems to give good results, as even at the end of the first hour we have some finished jobs in the last buffer. If we look at the buffers behaviour (Table 5), we can see that this result is obtained working three time slots.

Table 5  Buffers state

<table>
<thead>
<tr>
<th>Timeslot 0</th>
<th>Timeslot 1</th>
<th>Timeslot 2</th>
<th>Timeslot 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>b0</td>
<td>10</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>b1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b3</td>
<td>10</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>b4</td>
<td>3</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>b5</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b6</td>
<td>0</td>
<td>23</td>
<td>23</td>
</tr>
</tbody>
</table>

Case B

Here, we use the algorithm and ILOG solver to find the optimal scheduling; with these initial conditions (Table 3) and using all the other values, the algorithm completes all the job in the line in two hours. In Table 6, the related scheduling plan is shown.

This schedule gives the progress shown in Table 7 for the buffers state, with two hours required to process all the jobs (buffer b6 holds the finished works).
Table 6  Case B

<table>
<thead>
<tr>
<th>Timeslot 1</th>
<th>Timeslot 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ws1</td>
<td>Op1</td>
</tr>
<tr>
<td>Ws2</td>
<td>Op3</td>
</tr>
<tr>
<td>Ws3</td>
<td>Op1</td>
</tr>
<tr>
<td>Ws4</td>
<td>Op1</td>
</tr>
<tr>
<td>Ws5</td>
<td>Op2</td>
</tr>
<tr>
<td>Ws6</td>
<td>Op2</td>
</tr>
</tbody>
</table>

Table 7  Buffers state

<table>
<thead>
<tr>
<th>Timeslot 0</th>
<th>Timeslot 1</th>
<th>Timeslot 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>b0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>b1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b3</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>b4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>b5</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>b6</td>
<td>0</td>
<td>10</td>
</tr>
</tbody>
</table>

Comparison

Although in the first case in the first hour there are already finished jobs, the second solution given by ILOG is better. In Table 8, the most important performance parameters are shown.

Table 8  Parameters evaluation

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Case B</th>
<th>Case A</th>
</tr>
</thead>
<tbody>
<tr>
<td>Man-hours</td>
<td>6</td>
<td>9</td>
</tr>
<tr>
<td>Makespan (hours)</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Throughput (jobs/hour)</td>
<td>16.5</td>
<td>11</td>
</tr>
</tbody>
</table>

This significant improvement is obtained on a simple example with a little number of workstations and operators, in a full production line the results can have an high increase.

3.4  Case study implementation and results

In this section, we present the solution obtained for the industrial case. The line is supposed to be at the beginning of a production cycle, so the jobs are all in the first buffer. With the notations given in Section 3.1, the data related to the production line under analysis (Figure 2) are given in Table 9. The mathematical model is translated into 303 constraints and 256 variables (Figure 3). In addition to the data shown in Section 3 in the following Table 10, are specified the initial conditions of the buffers.
Figure 3  Data related to the mathematical model (see online version for colours)

Table 9  Problem data

<table>
<thead>
<tr>
<th>Opnum</th>
<th>Wnum</th>
<th>Sbuffers</th>
<th>Kshifts</th>
<th>λ</th>
<th>Γ</th>
<th>Φ</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>10000</td>
<td>1000</td>
<td>100</td>
</tr>
</tbody>
</table>

Table 10  Initial conditions of the buffers

<table>
<thead>
<tr>
<th>Buffers</th>
<th>b₀</th>
<th>b₁</th>
<th>b₂</th>
<th>b₃</th>
<th>b₄</th>
<th>b₅</th>
<th>b₆</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantity</td>
<td>140</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

As some example of programming, in Figure 4 the model of the objective function (3.2) is shown, in Figure 5 the implementation of constraint (3.5) and in Figure 6 the implementation of constraint (3.17).

Figure 4  Objective function (see online version for colours)

```latex
\text{maximize} \quad 10000 \cdot b_6 \text{maxline} \\
- \sum (k \text{ in shifts}) (x_{11}[k] + x_{21}[k] + x_{31}[k] + x_{12}[k] + x_{22}[k] + x_{32}[k] + x_{13}[k] + x_{23}[k] + x_{33}[k] + x_{14}[k] + x_{24}[k] + x_{34}[k] + x_{15}[k] + x_{25}[k] + x_{35}[k] + x_{16}[k] + x_{26}[k] + x_{36}[k]) \\
+ \sum (k \text{ in shifts}) (1000 - 1000 \cdot k) \times (p_{-1}[k] + p_{-2}[k] + p_{-3}[k] + p_{-4}[k] + p_{-5}[k] + p_{-6}[k]) \\
```

Figure 5  Buffer dynamic (see online version for colours)

```latex
\text{forall} (k \text{ in shifts}) \quad b_{-1}[k] \quad -- \quad b_{1}[k-1] \times p_{-1}[k] \quad - \quad p_{2}[k]:
```
In the following, the ILOG output gives the centralised scheduling shown in Table 11 and the buffers’ final conditions in Table 12.

**Table 11  Centralised scheduling**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ws1</td>
<td>Op2</td>
<td>Op1</td>
<td>Op2</td>
<td>Op1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ws2</td>
<td>Op1</td>
<td>Op3</td>
<td>Op1</td>
<td>Op3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ws3</td>
<td>Op2</td>
<td>Op1</td>
<td>Op2</td>
<td>Op1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ws5</td>
<td>Op1</td>
<td>Op2</td>
<td>Op3</td>
<td>Op3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ws6</td>
<td>Op2</td>
<td>Op1</td>
<td>Op1</td>
<td>Op2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 12  Buffers’ final state**

<table>
<thead>
<tr>
<th>Buffers</th>
<th>b0</th>
<th>b1</th>
<th>b2</th>
<th>b3</th>
<th>b4</th>
<th>b5</th>
<th>b6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quantity</td>
<td>16</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>104</td>
</tr>
</tbody>
</table>

The scheduling given by the algorithm has been compared with the data obtained in an experimental campaign in which a set of 30 roster-scheduling made by the operators themselves have been observed. The criteria to select the scheduling to be measured have been related to select those one with the same number of pieces to be produced, in the way to be able to compare the output of the main performance parameters.
The histogram of Figure 7 puts in comparison the parameters of Table 8 obtained through the optimisation algorithm with the mean value of the 30 measurements and the worst case occurred during this campaign.

There is a general improvement of all the indicators: among these, the best values can be recognised on WIP and on makespan, especially with respect to the worst case, while for every indicator the best cases are never equal or greater than the optimised ones.

Finally, with respect to the parameter man/hours and makespan, we can see that the best values are not far from those calculated by our algorithm.

For these two parameters we have verified that the improvement range is comprised between the 12% and 15% for the most part of the experiments made. Furthermore, during the experimental campaign we have verified that only the 3% of all the experiments shows values improved of less than 10% with respect to those given by the optimisation algorithm.

4 Conclusions

The present paper has been intended to front the problem of operator-scheduling on assembly lines where the number of workstations is greater than the number of operators. The chosen approach combines the principles of CSP and COP.

The main goal of this research work have been mainly related to an optimal utilisation of workforce, together with a maximisation of certain production parameters, such as throughput and makespan in an assembly line composed of six workstations.

With this aim, a mathematical model based on an objective function and 14 constraints has been developed. The model has been built in the way it can be applied to a wide set of CSP problems. The result has been successfully tested by a comparison with a man-based optimal solution.

The model has been then implemented for an industrial case, giving optimal results especially in terms of WIP decreasing and throughput increasing.

Regarding the results obtained, the conclusion of this work is that for certain productive systems characterised by several different tasks to be accomplished, the combined utilisation of COP and CSP techniques can be a key factor to obtain a complete solution of the problem.

The model developed in this paper can be improved including the possibility to manage uncertain events, such as machines breakdowns and/or operators unavailability’s, giving the system the possibility of a re-scheduling of resources with respect to the changed scenario. This kind of objective could be a further development of this work.

References


Flow shop operator scheduling


