Large-scale assembly job shop scheduling problems with bill of materials: models and algorithms

GIANPAOLO GHIANI, ANTONIO GRIECO, ANTONIO GUERRIERI
ANDREA MANNI, EMANUELE MANNI
Università del Salento
Dipartimento di Ingegneria dell'Innovazione
Via per Monteroni, 73100 Lecce
ITALY
{gianpaolo.ghiani, antonio.grieco, antonio.guerrieri, andrea.manni, emanuele.manni}@unisalento.it

Abstract: In this paper, we study an assembly job shop scheduling problem with tree-structured precedence constraints and jobs characterized by specific bills of materials. We propose a mathematical model to deal with a simplified version of the problem, as well as a fast and efficient constructive heuristic that is able to easily face real-world-sized instances. The production schedule takes into account the actual availability of materials in stock as well as the supply times and the capacity constraints, with the goal to minimize the average delay with respect to the due dates associated to the customers' orders. Computational results on data related to real-life instances show that the mathematical model is able to solve (not always to optimality) small-sized instances only. On the other hand, our heuristic approach is able to solve efficiently very large problems. Moreover, the proposed heuristic turns out to be scalable as the instance size grows.

Key–Words: Assembly job shop; Bill of materials; Operations scheduling; Heuristics

1 Introduction

Operations scheduling in job shop systems has received a lot of attention from the scientific community in the last decades. In the general job shop scheduling problem, n jobs (operations) must be scheduled on m available resources, satisfying a set of routing and factory constraints, with the objective of optimizing a given performance measure (typically, a function of the operations sequencing at each resource or, alternatively, the makespan). According to its production routine, each job is processed on machines with a given processing time, and each machine can typically process only one type of operation. However, in modern manufacturing plants, a machine may have the flexible capability to be set up to process more than one type of operations.

Relatively much less effort has been put on assembly job Shop (AJS) problems in which end-items are characterized by a bill of materials (BOM). The BOM generates complex operations precedence structures in the form of assembly trees with different levels. End-items are produced by means of sequences of fabrication and assembly operations and their components may have specific BOMs. A higher level item cannot be processed unless all preceding lower level items have been completely processed and assembled. AJS problems suffer from the same combinatorial explosion ill-effect of the "classical" job shop problem, for which significant research has been carried out. Although some authors have proposed exact methods (see, e.g., [5] or [4]), they are obviously not appropriate to deal with large problems, because of the NP-hardness of the problem. Thus, the attention of researchers has been devoted to the development of heuristic or metaheuristic algorithms with the aim to find good feasible solutions within a short computational time.

In this paper (that is an extension of [11]) we aim at devising both a mathematical model and a fast and efficient constructive heuristic for operations scheduling in AJS systems. Indeed, the need of a fast heuristic steams from two aspects. First, as we report in Section 6.2, the mathematical model (which is the result of a number of assumptions and simplifications) is able to deal with very small instances only. Second, when a salesman visits new possible customers, he/she must be able to get a quick and reliable response about the possibility to incorporate new orders without delaying the previously scheduled orders. In this sense, our procedure is able to easily deal with real-world-sized instances. The production schedule must take into account the actual availability of materials in stock as well as the supply times and the capacity constraints. The goal is to minimize the average delay with respect to the due dates associated to customers' orders.

The remainder of the paper is organized as follows. In Section 2 we review relevant literature. In Section 3 we provide the statement of the problem we study, whereas in Section 4 we propose a mathematical model to solve a simplified version of the original problem. In Section 5 we describe our heuristic solution approach and in Section 6 we illustrate our computational results on a number of instances resembling a real-life test case. Finally, concluding remarks follow in Section 7.

2 Literature review

Significant research has been done in the area of job shop scheduling. Although most versions of this problem are NP-hard ([20]), several authors have proposed exact methods based, for instance, on dynamic programming ([5], [12]) and branch and bound ([19], [2], [4], [24]).

Even some 'simplified versions' of the problem are still NP-Hard. Examples are problems with three machines and three jobs with an arbitrary number of operations per job (a job may have to visit a machine more than once) [3]; three machines and unitary processing times [3]; three machines and no more than two operations per job [20]; two machines and no more than three operations per job [20]. Some particular cases of the job-shop scheduling problem can be solved in polynomial time, such as the problem with two machines and no more than two operations per job [18] and the problem with two machines and unitary processing times [14].

One popular research direction is to solve the scheduling problem via heuristic or metaheuristic algorithms. These algorithms include the shifting bottleneck procedure ([1], [22], [41]), tabu search and simulated annealing ([32], [39], [31], [27]), memetic algorithms [13], genetic and evolutionary algorithms ([25], [29], [26]), artificial bee colony [37], neighborhood search algorithms [30], particle swarm optimization [21]. In [1] the authors propose an approximation method for solving the job shop scheduling problem by minimizing the makespan. Machines are sequenced one at a time by selecting at each iteration, among the machines not sequenced yet, the one classified as a bottleneck. Afterwards, all previously defined sequences are locally reoptimized. Both the bottleneck identification and the local reoptimization procedures are based on repeatedly solving one-machine scheduling problems. In [22] a hybrid shifting bottleneck procedure (HSBP) algorithm combined with Tabu Search (TS) is developed to deal with the parallel-machine job-shop scheduling problem. The authors propose an algorithm to decompose the parallel-machine job-shop scheduling problem in a set of single-machine scheduling and/or parallel-machine scheduling subproblems. [41] consider job shop scheduling problems with transportation constraints and bounded processing times. They represent the characteristics and constraints of the considered problem by means of a modified disjunctive graph and use a modified shifting bottleneck procedure to solve the problem. [32] describe an approximation algorithm based on Simulated Annealing (SA) for the problem of identifying the solution with the minimum makespan in a job shop system. [39] develop a heuristic search approach combining SA and Tabu Search (TS). The basic idea is first to use SA to find promising solutions, and then to employ TS to improve such solutions. For the same problem, [21] define a hybrid algorithm consisting of particle swarm optimization, SA and a multi-type individual enhancement scheme. [27] present an algorithm incorporating TS into the framework of path relinking to generate solutions to the job shop scheduling problem. This algorithm comprises a relinking procedure to effectively construct a path linking, as well as a reference solution determination mechanism. [31] address the classic job shop scheduling problem with sequence dependent setup times and develop a TS algorithm with a sophisticated neighborhood structure. [13] describe a memetic algorithm for solving a job shop scheduling problem. They propose three priority rules, (partial reordering, gap reduction and restricted swapping) and use a local search algorithm. [29] design some genetic operators. To increase the diversity of the population, they give a mixed selection operator based on the fitness value and the concentration value. Moreover, in order to make full use of the characteristics of the problem, they design a new crossover operator based on the machine and mutation operator.

In addition to the basic job shop scheduling problem, some authors have also studied its flexible variant. [10] propose an improved GA to solve the Distributed and Flexible Job-shop Scheduling problem. They make use of a new local-search-based operator to improve available solutions by refining the most promising individuals of each generation. [36] devise a parallel variable neighborhood search algorithm based on the application of multiple independent searches to improve the exploration of the search space. [40] study a problem similar to [36] and propose an effective GA with the goal of minimizing the makespan. [7] face the flexible job shop scheduling problem with parallel machines and reentrant process. Their procedure is based on a GA and a Grouping GA and consists of two main modules: a machine selection module (MSM) and an operation scheduling module (OSM). MSM helps an operation to select one of the parallel machines to process the job, whereas OSM is then used to arrange the sequences of all operations assigned to each machine. [34] address a multiobjective flexible job-shop scheduling problem with three objectives: minimizing the makespan, the total workload, and the maximum workload. For this problem the authors develop a hybrid multiobjective evolutionary approach. More recently, [42] focus on the flexible job shop scheduling problem with uncertainties due to urgently arrival jobs, working condition of the machines, etc. They propose an inserting algorithm and use condition-based maintenance to reduce machines unavailability.

As far as AJS scheduling problems are concerned, earlier contributions are due to [9] and [8]. The former authors consider the presence of BOM and face the issue of integrally scheduling parts that are related through a BOM. The goal is to improve the on-time performance of products as well as reducing work-in-process inventory, by employing a technique based on Lagrangian relaxation (LR). [8] propose a constructive heuristic for minimizing the production time of jobs involving both machining and assembly operations in a production shop. [38] analyze the requirements of a real-life AJS scheduling problem. First, the authors model the problem by means of timed colored Petri nets. Then, they solve it in a Branch and Bound fashion. [28] devise a new heuristic method for simulating and supporting the operations scheduling process in AJS systems. The method is based on the assumption that an improvement in operations synchronization at fabrication and assembly stations allows forth better achievement of due dates. The method implements two scheduling approaches: a backward approach satisfying due date completely and a forward approach satisfying capacity restrictions completely. The two approaches work iteratively within two different simulation models of the production system, a deterministic one and a probabilistic one. [23] deal with an AJS scheduling problem for which the propose an integrated application of order review/release mechanism and dispatching rules. [35] consider an assembly scheduling problem with tree-structured precedence constraints and propose a mixed-integer linear programming formulation solved with a LR approach. [33] develop a hybrid GA and a hybrid particle swarm optimization procedure to minimize the makespan of a job shop scheduling problem including the assembly stage.

The main contribution of this paper to the existing literature is twofold. First, we introduce a new version of the AJS scheduling problem with some peculiar constraints. Second, we present a fast and effective constructive heuristic procedure which is able to deal with real-world-sized instances, with up to 6500 jobs to be scheduled and 400 work centers. As a minor contribution, we propose a mathematical model to deal with a simplified version of the problem.

3 Problem statement

In this paper, we consider an AJS system involving a manufacturing facility with U units (e.g., cutting, bending, machining, welding) and W work centers. Each unit $u = 1, \dots, U$ is made up of W_u work centers (e.g., the cutting unit can host work centers for plasma cutting, laser cutting, or oxifuel cutting), and each work center $w = 1, \dots, W$ consists of m_w functionally identical machines. Given an end product (which could eventually require to be produced at a known lot size), its Bill-of-Materials (BOM) defines the assembly relationships between components and has a tree-like precedence structure with a number of levels. Each node of the tree represents a part that must be produced or bought in order to realize the end product. Figure 1 reports a sample BOM, in which there are four levels (level 0 represents the end product). In this figure, the end product requires four parts in order to be assembled. In turn, each part of level 1 (except for node 2) has its own BOM, requiring other parts in order to be manufactured, and so on.
We make the following assumptions:

- a machine can perform at most one operation at a time;
- an operation can be performed on at most one machine at a given time;

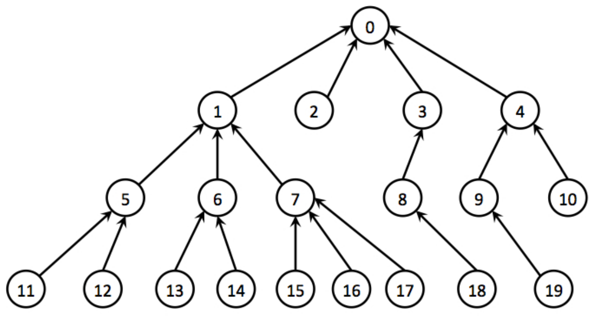

Figure 1: A sample Bill-of-Materials

- processing time of all operations are deterministic and known;
- several production phases could be needed in order to produce a part; in this case, each phase is characterized by a work center and a processing time;
- backlogging is permitted, in that some machines could have some operations to complete (i.e., their capacity is reduced by the sum of the processing times of such operations) at the beginning of the planning horizon.

Before presenting the heuristic approach we devise for operations scheduling in AJS systems, we introduce the following further notation:

- T is the planning horizon;
- e_w is the machines' productivity in work center w. This parameter is used to reduce the capacity of work centers in case of known downtimes such as those due to maintenance operations;
- s_{wt} is the number of shifts of work center w in the day t;
- Q_{wjt} is the residual capacity of work center w for shift j in the day t;
- D is the set of customers' demands;
- τ_d is the due date of demand d;
- q_d is the quantity required by demand d;
- ID_d is the identifier of the end product required by demand d;
- a_d is the priority of demand d;
- g_d is the number of days to transport the end product of demand d from the manufacturing facility to the customer;
- P is the set of parts;
- O_p is the set of purchase orders already issued for part p;
- r_o is the ready time of purchase order o;
- n_o is the number of parts that will be delivered with purchase order o;
- I_p is the inventory level of part p;
- C_p is the set of phases for producing part p;
- mt_c is the machine time for production phase c;
- w_c is the work center for production phase c;
- UM_p is the unit of measure of the lot for part p;
- l_p is the lot size for part p;
- ml_p is the size of multiple lot for part p;
- BOM$_p$ is the BOM of part p;
- F_p is the set containing the children of part p in the BOM;
- for each part p and for each $f \in F_p$, b_f is the number of pieces of f to be mounted on the father p;
- av_p is the average number of units of part p utilized daily;
- L_{BOM_p} is the number of levels in the BOM of part p;
- B_{ip} is the set of parts of level i in the BOM of part p.

The decisions aspects related to the proposed approach involve the possibility to satisfy customers' demands in several ways. First, if a demand can be satisfied with the available stock of end-product, there is no need to schedule anything. Otherwise, for each node of each level of the BOM we must decide whether the requirements of the various components can be satisfied with the stock or we must schedule their production. In the former case, we do not need to decide anything for the node and all its children along the current branch of the BOM tree. In the latter case, we must decide how much to produce, taking into account policies involving setup minimization, or requirements about the production with minimum or multiple lots. Furthermore, when deciding to produce a given quantity of a component, it is crucial to identify the time slots in which to allocate each of its production phases.

4 A mathematical model

In this section, we present a mathematical model for our assembly job shop problem. However, due to the complexity of the whole problem, it was necessary to make some simplifying assumptions in order to make it tractable. In particular:

- the mathematical model is focused on a high-level scheduling at the departments level in place of a detailed schedule at the work centers level;
- the model provides the scheduling of end products in place of parts;
- the processing of an end product in a department is completed in a working day.

Before presenting the mathematical model, we introduce the following further notation:

- DEP is the set of company departments;
- DEP_d is the set of departments where the end product required by demand d must be realized;
- A_r is the capacity of department r;
- a_{dr} is the time needed to realize the end product of demand d in the department $r \in DEP_d$.

Thus, a mathematical formulation, that minimizes the overall delay for customers' demand, is:

$$(P) \quad \min \sum_{d \in D} \sum_{r \in DEP_d} \sum_{t \geq \tau_d} (t - \tau_d) x_{drt} \tag{1}$$

$$\text{s.t.} \quad \sum_{d \in D : r \in DEP_d} \sum_{t=1}^{T} a_{dr} q_d x_{drt} \leq A_r, \quad \forall r \in DEP \tag{2}$$

$$\sum_{r \in DEP_d} x_{drt} \leq 1, \;\; \forall d \in D, \forall t \in T \tag{3}$$

$$\sum_{t \in T} x_{drt} = 1, \;\; \forall d \in D, \forall r \in DEP_d \tag{4}$$

$$x_{dr't'} \leq 1 - x_{drt}, \;\; \forall d \in D; \;\; \forall r, r' \in DEP_d : r' > r; \;\; \forall t, t' \in T : t' \leq t;. \tag{5}$$

$$x_{drt} \in \{0, 1\}, \;\; \forall d \in D, \forall r \in DEP_d, \forall t \in T. \tag{6}$$

Here, variables x_{drt} ($d \in D$, $r \in DEP_d$, $t \in T$) are binary variables taking value 1 if the end product required by demand d is processed in department r in working day t. Inequalities (2) represent capacity constraints at the department level. In particular, they impose that the products processed in a working day in a specific department do not exceed the total capacity of the department. Constraints (3) ensure that an end product is processed, at most, by a single department on a single working day. Constraints (4) force the processing of an end product in a department to be completed in a single working day. Constraints (5) require that the machining operations are carried out in sequence starting from the first available working day. Moreover, a machining operation cannot be carried out before the previous one is completed. Finally, constrains (6) define the domain of the variables.

5 A fast heuristic

The goal of our constructive heuristic procedure is to define efficient and feasible production plans, taking into account the actual availability of materials at stock as well as the supply times. The production plan must also satisfy capacity constraints related to the availability of work centers. Moreover, we aim at meeting the due dates of customers' demands as much as possible. The basic idea is described in Algorithm 1. First of all, the demands are sorted with respect to some criteria (e.g., with respect to due dates, priorities, or a combination of both). Then, for each demand d, we reduce the quantity q_d by taking into account stocks, and we update the inventory level (lines 4 and 5). If q_d is still positive, we determine the quantity of end product to be realized, by taking into account lots (line 7). Then, we retrieve the structure of the BOM for end product ID_d and we prune it, by removing branches of the tree that do not need to be produced because of inventory availability (lines 8 and 9, CONSTRUCTBOM and PRUNE procedures). Finally, on the basis of the pruned BOM, we try to allocate the nodes of the BOM to the work centers (line 10). If this operation succeeds, we update the solution (line 12), and consider the next demand.

Algorithm 2 describes how to construct the BOM, by taking into account the identifier of the end product

Algorithm 1 A constructive heuristic for operations scheduling in AJS systems

```
1: procedure SCHEDULE($W$, $T$, $D$, $P$)
2:     < sort demands in $D$ >
3:     for all $d \in D$ do
4:         $q_d \leftarrow \max(0, q_d - I_{ID_d})$
5:         < update $I_{ID_d}$ >
6:         if $q_d > 0$ then
7:             $q_d \leftarrow$ COMPUTEQ($ID_d, UM_{ID_d}, l_{ID_d}, ml_{ID_d}, q_d$)
8:             BOM$_{ID_d} \leftarrow$ CONSTRUCTBOM($ID_d, q_d$)
9:             BOM$_{ID_d} \leftarrow$ PRUNE(BOM$_{ID_d}$)
10:            result $\leftarrow$ ALLOCATEDEMAND(BOM$_{ID_d}$)
11:            if result == true then
12:                < update the solution >
13:            end if
14:        end if
15:    end for
16: end procedure
```

Algorithm 2 The procedure for constructing BOM

```
1: procedure CONSTRUCTBOM($ID_d$, $q_d$)
2:     BOMtree $\leftarrow$ null
3:     node $\leftarrow$ null
4:     node.id $\leftarrow ID_d$
5:     node.quantity $\leftarrow q_d$
6:     < set node as the root of the BOM tree >
7:     BOMtree $\leftarrow$ FINDCHILDREN(node, BOMtree)
8:     return BOMtree
9: end procedure
```

Algorithm 3 The procedure for finding the children of a node in the BOM

```
1: procedure FINDCHILDREN(node, BOMtree)
2:     for all $p \in F_{\text{node.id}}$ do
3:         childNode $\leftarrow$ null
4:         childNode.id $\leftarrow p$
5:         childNode.quantity $\leftarrow$ node.quantity $* b_p$
6:         < add childNode as child of node in BOMtree >
7:         BOMtree $\leftarrow$ FINDCHILDREN(childNode, BOMtree)
8:     end for
9:     return BOMtree
10: end procedure
```

and the quantity required by demand d. In particular the BOM tree and the current node are initially set to null (lines $2 - 3$). Then we set the identifier and the quantity required by the node (lines $4 - 5$) and the current node is set as the root of the actual BOM tree (line 6). Next, we construct the BOM tree by using Algorithm 3 (line 7), and we output it (line 8).

Algorithm 3 is a recursive procedure for finding the children of a node in the BOM. In particular, for each child of a given end product or part, we set the id and the quantity of the current child node (lines $4 - 5$). Then, we add the current child node to the BOM tree (line 6). Next, we find the children nodes of the current node by recursively calling the procedure (line 7). Finally, we output the BOM tree (line 9).

In Algorithm 4 we illustrate the procedure for computing the amount of end product to be produced, by taking into account the quantity required by the customer and the information about lots. In particular, the lot size is determined differently depending on the value of the "unit of measure" parameter. If the unit of measure is "days", then the minimum lot to produce is equal to the overall demand of product ID_d in the subsequent l_{ID_d} days. If the unit of measure is "quantity", then the minimum lot size is obtained as the maximum between q_d and l_{ID_d}. If the unit of measure is "units", then the minimum lot size is computed on the basis of the average daily consumption of the product (line 8). Finally, the actual quantity to produce is determined according to the minimum lot size and the information about the need to produce the part in multiples of a given value (Algorithm 5).

Algorithm 6 describes how to prune the BOM tree, by reducing the amount to be produced at each node on the basis of the actual inventory levels. For each level i of the BOM and for each part p to be produced at level i, we consider the number of pieces of p to be mounted on its father. If the quantity b_p is at stock, then we remove p from level i as well as all its children along the branch of the BOM and we update the inventory level of p (lines $4 - 6$). Otherwise, we reduce the quantity b_p by the inventory level I_p, and we determine the new quantities b_p of this branch of the BOM tree (lines $7 - 10$).

Algorithms 7 and 8 represent the core of our constructive procedure for operations scheduling. In Algorithm 7, by iterating on the levels of the BOM, for each part p we first update the quantity b_p in order to meet lots requirements (line 9). Then, we check whether there are purchase orders already issued, and update quantities b_p and the day t when the job is scheduled, according to ready times and quantities of the purchase orders (lines $10 - 20$). Next, if b_p is still positive, we allocate each production phase of part p, by means of the ALLOCATEPHASE procedure (lines $21 - 28$).

The procedure for allocating each production phases is described in Algorithm 8. The goal is to identify couples day/shift in order to assign the job to an appropriate work center. An important assumption is that a job may be split among consecutive shifts. In

Algorithm 4 The procedure for computing the quantity of end product to be realized

```
1: procedure ComputeQ($ID_d, UM_{ID_d}, l_{ID_d}, ml_{ID_d}, q_d$)
2:     minumumLot $\leftarrow 0$
3:     if $UM_{ID_d} ==$ "days" then
4:         minimumLot $\leftarrow$ $<$ find the need of end product $ID_d$ in the next $l_{ID_d}$ days $>$
5:     else if $UM_{ID_d} ==$ "quantity" then
6:         minimumLot $\leftarrow \max(q_d, l_{ID_d})$
7:     else if $UM_{ID_d} ==$ "units" then
8:         minimumLot $\leftarrow \max(q_d, l_{ID_d} * av_{ID_d})$
9:     end if
10:    $q \leftarrow$ ComputeMultiple(minimumLot, $ml_{ID_d}$)
11:    return $q$
12: end procedure
```

Algorithm 5 The procedure for determining the actual lot size

```
1: procedure ComputeMultiple(minimumLot, $ml_{ID_d}$)
2:     $q \leftarrow 0$
3:     if $ml_{ID_d} \geq$ minimumLot then
4:         $q \leftarrow ml_{ID_d}$
5:     else
6:         $q \leftarrow$ $<$ find the smallest multiple of $ml_{ID_d}$ that is not lower than minimumLot $>$
7:     end if
8:     return $q$
9: end procedure
```

fact, if the residual capacity of the work center w_c under consideration is enough to accommodate the job, we are done (lines $6-10$). Otherwise, we allocate to w_c a job involving a fraction of b_p computed as $\lfloor \frac{\text{residualCapacity}}{mt_c} \rfloor$, and we update b_p and the machine time needed to perform the job (lines $11-15$). Then, we seek for another shift of the same day (or, alternatively, another working day) to assign the reduced value of b_p. The procedure iterates until the whole quantity b_p is allocated or we reach the end of the planning horizon.

6 Computational results

In this section, we test the performance of our mathematical model and our heuristic on a set of real-life instances, related to a manufacturing company. The approach is coded in Java and is tested on a machine equipped with an Intel i7 processor clocked at 3.0 GHz and 8 GB of RAM. In order to solve the optimization models, we use the general-purpose black-box MIP solver IBM ILOG CPLEX 12.5 [17].

Algorithm 6 The procedure for pruning the BOM

```
1: procedure Prune(BOM$_{ID_d}$)
2:     for all $i = 0, \ldots, L_{\text{BOM}_{Id_d}}$ do
3:         for all $p \in B_{iID_p}$ do
4:             if $b_p < I_p$ then
5:                 $<$ remove $p$ from level $i$ as well as all its children along this branch $>$
6:                 $I_p \leftarrow I_p - b_p$
7:             else
8:                 $b_p \leftarrow b_p - I_p$
9:                 $<$ update the quantities for all the children of $p$ along this branch $>$
10:                $I_p \leftarrow 0$
11:            end if
12:        end for
13:    end for
14:    return BOM$_{ID_d}$
15: end procedure
```

6.1 Test problem

To test our approach, we use, as a benchmark, the current situation of a manufacturing company. The company consists of several departments such as cutting, bending, machining, welding, painting, testing. In particular in the cutting department there are 23 work centers using two cutting technologies (laser and plasma). The bending department is composed of 29 working center with robots. The machining department uses 32 horizontal and vertical computer numerical control work centers. In the welding department there are 45 robots. The painting department uses two different painting technologies (primer painting and wet top coat) and is composed of 4 painting systems with 12 work centers for the final assembly of all items. Finally, the testing department performs a three-dimensional control of measures and ultrasonic tests for welding.

The instances considered for testing have up to 4300 jobs to be scheduled on about 400 work centers. The maximum number of production phases is two. The average number of levels in the BOM of end products is 11 (the maximum number of levels in the BOM is 33, while the minimum number of levels in the BOM is two).

6.2 Preliminary results for the optimization model

In this section we report the results of some preliminary tests performed on the mathematical model of Section 4. In particular, we have considered 5 instance

Algorithm 7 The procedure for allocating demand

```
1: procedure AllocateDemand(BOM$_{ID_d}$)
2:     $t \leftarrow 1$
3:     result $\leftarrow$ true
4:     for all $i = L_{\mathrm{BOM}_{ID_d}}, \dots, 0$ do
5:         if result == false then
6:             break
7:         end if
8:         for all $p \in B_{iID_d}$ do
9:             $b_p \leftarrow$ ComputeQ$(p, UM_p, l_p, ml_p, b_p)$
10:            for all $o \in O_p$ do
11:                if $n_o \geq b_p$ then
12:                    $n_o \leftarrow n_o - b_p$
13:                    $b_p \leftarrow 0$
14:                    $t \leftarrow r_o$
15:                    break
16:                else
17:                    $b_p \leftarrow b_p - n_o$
18:                    $n_o \leftarrow 0$
19:                end if
20:            end for
21:            if $b_p > 0$ then
22:                for all $c \in C_p$ do
23:                    result $\leftarrow$ AllocatePhase$(c, t, p, b_p)$
24:                    if result == false then
25:                        break
26:                    end if
27:                end for
28:            end if
29:        end for
30:    end for
31:    return result
32: end procedure
```

Algorithm 8 The procedure for allocating production phases

```
1: procedure AllocatePhase$(c, t, p, b_p)$
2:     machineT $\leftarrow mt_c * b_p$
3:     while $b_p > 0$ && $t \leq T$ do
4:         for all $j = 1, \dots, s_{w_c t}$ do
5:             residualCapacity $\leftarrow Q_{w_c jt}$
6:             if residualCapacity $\geq$ machineT then
7:                 < allocate to work center $w_c$ a job for the
                   production of $b_p$ units of $p$, taking $mt_c * b_p$
                   time units >
8:                 < update the residual capacity of $w_c$ >
9:                 $b_p \leftarrow 0$
10:                break
11:            else
12:                < allocate to work center $w_c$ a job for
                   the production of $\lfloor \frac{\text{residualCapacity}}{mt_c} \rfloor$ units of $p$,
                   taking $\lfloor \frac{\text{residualCapacity}}{mt_c} \rfloor * mt_c$ time units >
13:                $b_p \leftarrow b_p - \lfloor \frac{\text{residualCapacity}}{mt_c} \rfloor$
14:                machineT $\leftarrow$ machineT$- \lfloor \frac{\text{residualCapacity}}{mt_c} \rfloor * mt_c$
15:            end if
16:        end for
17:        if $b_p > 0$ then
18:            $t \leftarrow t + 1$
19:        end if
20:    end while
21:    if $b_p == 0$ then
22:        return true
23:    else
24:        return false
25:    end if
26: end procedure
```

types, each of which is characterized by a different number of customers requests (between 25 and 100) as well as a different length of the planning horizon (between 2 months and 4 months). For each instance type we have considered 10 instances. For each instance, we set a maximum time limit of 3,600 seconds. The results are presented in Table 1. First of all, we observe that not every instance type can be solved to optimality (this happens for the instances having a non-zero value in the column 'Optimality gap'). In this case, the average optimality gap is about 23%. With respect to the average delay, the (optimal or sub-optimal) solutions provided by the model have an average value of 13.31 days, with a maximum delay of about 20 days and a minimum delay of about 7 days. In addition, on the average about 47% of the customers' demands are scheduled before their due date.

As can be observed from the table, as the instance size grows, it becomes more and more difficult for the model to be solved to optimality. Thus, for real-world-sized instances it is fundamental to employ a fast and effective heuristic to face the problem.

6.3 Results for the heuristic

Table 2 reports the average results obtained by the heuristic on 5 instance types, characterized by a different number of customers requests as well as a different length of the planning horizon. These characteristics are reported in column 'Instance features'. For each instance type (and hence for each line of the table), we have considered 10 instances. The average time needed to schedule a single request is about 0.02 seconds. As emerges from the table, the heuristic is scalable as the instance size grows. The average delay for customers requests is 0.47 days, with a maximum delay of about six days and a minimum (negative) delay of about three days. A negative delay means that the orders are scheduled to be produced before the due date. Moreover, about 43% of the orders are sched-

Table 1: Results of the tests performed on the mathematical model.

Instance features ($\|D\|, T$)	Time (sec)	Optimality gap (%)	Average delay (days)	Orders without delay (%)
$(25, 2)$	1,789	–	7.03	64.18
$(40, 2)$	2,864	–	9.18	51.96
$(50, 3)$	3,600	10.12	14.45	37.63
$(75, 3)$	3,600	19.41	15.87	39.77
$(100, 4)$	3,600	39.18	20.04	41.35

Table 2: Results of the tests performed on real-world-sized instances.

Instance features	Time	Average delay	Orders without	Average departments saturation (%)			
($\|D\|, T$)	(sec)	(days)	delay (%)	1 month	2 months	3 months	4 months
$(4067, 20)$	79.59	3.59	44.09	62.74	59.54	57.38	55.22
$(3284, 15)$	62.98	6.36	37.02	66.95	63.29	60.28	58.75
$(3981, 18)$	73.50	-2.64	38.69	61.83	62.16	59.26	57.24
$(3946, 15)$	80.19	-2.15	45.90	63.50	63.24	59.82	56.62
$(4131, 14)$	77.70	-2.81	47.93	61.34	60.35	59.10	56.84

Table 3: Results obtained by using the solutions provided by the company.

Instance features	Average delay	Orders without	Average departments saturation (%)			
($\|D\|, T$)	(days)	delay (%)	1 month	2 months	3 months	4 months
$(4067, 20)$	16.47	31.27	47.91	46.49	40.47	38.16
$(3284, 15)$	13.83	26.72	50.73	47.11	45.99	42.52
$(3981, 18)$	9.18	25.14	45.69	44.52	42.57	41.48
$(3946, 15)$	11.31	33.71	46.56	44.74	43.78	41.29
$(4131, 14)$	10.14	31.18	45.14	43.56	43.20	39.22

uled within the due date, on the average. Finally, another important statistics concerns the average saturation of departments. In particular, we have computed these data with reference to the first four months of scheduling, obtaining average values of about 63%, 62%, 59% and 57%, respectively. Apparently, these results could seem not completely satisfactory. Indeed, they are influenced by the presence of departments with a low workload. This would suggest to the company the need of a partial reconfiguration of its internal structure. In particular, this could be performed by moving both human and material resources from such under-utilized departments to those that are completely saturated and thus represent bottlenecks.

In order to provide additional insights about the effectiveness of our approach, in Table 3 we present the results obtained by using the "manual" solutions provided by the company on the same instances reported in Table 2. In this case, the average delay per request is 12.19 days, whereas the orders scheduled within the due date is about 38%, on the average. Moreover, the average saturation of departments is about 47%, 45%, 43% and 40%, for the first four months of scheduling, respectively. Finally, it is worthwhile to mention that it takes several working days to obtain such "manual" solutions.

7 Conclusions

In this paper we have studied an assembly job shop scheduling problem in which end-products are characterized by complex bills of materials. For this problem, we have proposed a mathematical model (to face a simplified version of the problem), as well as a fast and efficient constructive heuristic that takes into account the actual availability of materials at stock as well as the order due dates. Moreover, the production plans must also satisfy capacity constraints related to the availability of work centers as well as meet the due dates of customers' demands as much as possible. Computational results on data related to a real-life test case have shown that our approach is able to easily deal with instances with up to 4200 jobs to be scheduled on 390 work centers over a 20 months planning horizon. The effectiveness of our heuristic is attested by an average delay for customers requests of 0.47 days (12.19 days for the "manual" solution), with about 43% of the orders that are scheduled within the due date (38% for the "manual" solution). Another important remark concerns the computing time of our heuristic procedure. Indeed, the average time needed to schedule a single request is about 0.02 seconds. More important, this time remains approximately constant as the size of the instances grows.

Acknowledgements: The testing phase of the scheduling algorithm was partially supported by the Ministero dell'Istruzione, dell'Università e della Ricerca Scientifica (MIUR) of Italy under the project "VIS4FACTORY - Visual Information System for Factory" (PON02_00634_3551288) which aimed at integrating the scheduling algorithm into a production monitoring system based on artificial vision. This support is gratefully acknowledged.

References:

[1] J. Adams, E. Balas and D. Zawack. The shifting bottleneck procedure for job shop scheduling, *Management Science*, Vol. 34, Issue 3, 1988, pp. 391–401.

[2] D. Applegate and W. Cook. A computational study of job-shop scheduling, *ORSA Journal on Computing*, Vol.3, Issue 2, 1991, pp. 149–156.

[3] P. Brucker. Scheduling algorithms (2nd ed.), *Springer-Verlag New York, Inc*, Secaucus, NJ, USA (1998).

[4] P. Brucker, B. Jurisch and B. Sievers. A branch and bound algorithm for the job-shop scheduling problem, *Discrete Applied Mathematics*, Vol. 49, Issues 1–3, 1994, pp. 107–127.

[5] J. Carlier and E. Pinson. An algorithm for solving the job-shop problem, *Management Science*, Vol. 35, Issue 2, 1989, pp. 164–176.

[6] K.-P. Chen, M.S. Lee, P.S. Pulat and S.A. Moses. The shifting bottleneck procedure for job-shops with parallel machines, *International Journal of Industrial and Systems Engineering*, Vol. 1, 2006, pp. 244–262.

[7] J.C. Chen, C.-C. Wu, C.-W. Chen and K.-H. Chen. Flexible job shop scheduling with parallel machines using genetic algorithm and grouping genetic algorithm, *Expert Systems with Applications*, Vol. 39, Issue 11, 2012, pp. 10016–10021.

[8] D.H. Cummings and Mckoy P.J. Egbelu. Minimizing production flow time in a process and assembly job shop, *International Journal of Production Research*, Vol. 38, Issue 8, 1998, pp. 2315–2332.

[9] C.S. Czerwinski and P.B. Luh. Scheduling products with bills of materials using an improved lagrangian relaxation technique, *IEEE Transactions on Robotics and Automation*, Vol. 10, Issue 2, 1994, pp. 99–111.

[10] L. De Giovanni and F. Pezzella. An improved genetic algorithm for the distributed and flexible job-shop scheduling problem, *European Journal of Operational Research*, Vol. 200, Issue 2, 2010, pp. 395–498.

[11] G. Ghiani, A. Grieco, A. Guerrieri, A. Manni and E. Manni. A fast heuristic for large-scale assembly job shop scheduling problems with bill of materials, *Advances in Mathematics and Statistical Sciences*, Vol. 40, 2015, pp. 216–223.

[12] J.A.S. Gromicho, J. J. van Hoorn, F. Saldanha-da-Gama and G. T. Timmer. Solving the job-shop scheduling problem optimally by dynamic programming, *Computers & Operations Research*, Vol. 39, Issue 12, 2012, pp. 2968–2977.

[13] S.M.K. Hasan, R. Sarker and D. Essam. Memetic algorithms for solving job-shop scheduling problems, *Memetic Computing*, Vol. 1, Issue 1, 2009, pp. 69–83.

[14] N. Hefetz and I. Adiri. An efficient optimal algorithm for the two-machines unit-time jobshop schedule-length problem, *Mathematics of Operations Research*, Vol. 7, Issue 3, 1982, pp. 354–360.

[15] H. Hu and Z. Li. Modeling and scheduling for manufacturing grid workflows using timed petri nets, *The International Journal of Advanced Manufacturing Technology*, Vol. 42, Issue 5–6, 2009, pp. 553–568.

[16] H. Hu and Z. Li. Synthesis of liveness enforcing supervisor for automated manufacturing systems using insufficiently marked siphons, *Journal of Intelligent Manufacturing*, Vol. 21, Issue 4, 2010, pp. 555–567.

[17] IBM ILOG CPLEX. www.ibm.com/software/products/ibmilogcpleoptistud. Last accessed on 24th March 2015.

[18] J.R. Jackson. Scheduling a production line to minimize maximum tardiness, *Technical report, Management Science Research Project*, University of California, Los Angeles, 1955.

[19] B.J. Lageweg, J.K. Lenstra and A.H.G. Rinnooy Kan. Job-Shop Scheduling by Implicit Enumeration, *Management Science*, Vol. 4, 1979, pp. 441–450.

[20] J.K. Lenstra. Computational Complexity of Discrete Optimization Problems, *Annals of Discrete Mathematics*, Vol. 24, Issue 4, 1977, pp. 121–140.

[21] T.-L. Lin, S.-J. Horng, T.-W. Kao, Y.-H. Chen, R.-S. Run, R.-J. Chen, J.-L. Lai and I-H. Kuo. An efficient job-shop scheduling algorithm based on particle swarm optimization, *Expert Systems with Applications*, Vol. 37, Issue 3, 2010, pp. 2629–2636.

[22] S.Q. Liu and E. Kozan. A hybrid shifting bottleneck procedure algorithm for the parallel-machine job-shop scheduling problem, *Journal of the Operational Research*, Vol. 63, 2012, pp. 168–182.

[23] H.L. Lu, G.Q. Huang and H.D. Yang. Integrating order review/release and dispatching rules for assembly job shop scheduling using a simulation approach, *International Journal of Production Research*, Vol. 49, Issue 3, 2011, pp. 647–669.

[24] P. Martin and D.B. Shmoys. A new approach to computing optimal schedules for the job-shop scheduling problem, *Lecture Notes in Computer Science*, Vol. 1084, 1996, pp. 389–403.

[25] K. Mesghouni, S. Hammadi and P. Borne. Evolutionary Algorithms for Job-Shop Scheduling, *International Journal of Applied Mathematics and Computer Science*, Vol. 14, Issue 1, 2004, pp. 91–103.

[26] N. H. Moin, O. C. Sin and M. Omar. Hybrid Genetic Algorithm with Multiparents Crossover for Job Shop Scheduling Problems, *Mathematical problems in engineering*, Vol. 2015, Article ID 210680, 12 pages, 2015.

[27] B. Peng, Z. Lü and T.C.E. Cheng. A tabu search/path relinking algorithm to solve the job shop scheduling problem, *Computers & Operations Research*, Vol. 53, 2015, pp. 154–164.

[28] M.T. Pereira and M.C. Santoro. An integrative heuristic method for detailed operations scheduling in assembly job shop systems, *International Journal of Production Research*, Vol. 49, Issue 20, 2011, pp. 6089–6105.

[29] R. Qing-dao-er-ji and Y. Wang. A new hybrid genetic algorithm for job shop scheduling problem, *Computers & Operations Research*, Vol. 39, Issue 10, 2012, pp. 2291–2299.

[30] V. Roshanaei, B. Naderi, F. Jolai and M. Khalili. A variable neighborhood search for job shop scheduling with set-up times to minimize makespan, *Future Generation Computer Systems*, Vol. 25, Issue 6, 2009, pp. 654–661.

[31] L. Shen. A tabu search algorithm for the job shop problem with sequence dependent setup times, *Computers & Industrial Engineering*, Vol. 78, 2014, pp. 95–106.

[32] P.J.M. van Laarhoven, E.H.L. Aarts and J.K. Lenstra. Job shop scheduling by simulated annealing, *Operations Research*, Vol. 40, Issue 1, 1992, pp. 113–125.

[33] T.C. Wong and S.C. Ngan. A comparison of hybrid genetic algorithm and hybrid particle swarm optimization to minimize makespan for assembly job shop, *Applied Soft Computing*, Vol. 13, Issue 3, 2013, pp. 1391–1399.

[34] J. Xiong, X. Tan, K.-W. Yang, L.-N. Xing and Y.-W. Chen. A Hybrid Multiobjective Evolutionary Approach for Flexible Job-Shop Scheduling Problems, *Mathematical problems in engineering*, Vol. 2012, Article ID 478981, 27 pages, 2012.

[35] J. Xu and R. Nagi. Solving assembly scheduling problems with tree-structure precedence constraints: A lagrangian relaxation approach, *IEEE Transactions on Automation Science and Engineering*, Vol. 10, Issue 3, 2013, pp. 757–771.

[36] M. Yazdani, M. Amiri and M. Zandieh. Flexible job-shop scheduling with parallel variable neighborhood search algorithm, *Expert Systems with Applications*, Vol. 37, Issue 1, 2010, pp. 678–687.

[37] M. Yin, X. Li and J. Zhou. An efficient job shop scheduling algorithm based on artificial bee colony, *Scientific Research and Essays*, Vol. 5, 2011, pp. 2578–2596.

[38] D. Zang, X. Shi and M. Jin. Modeling and scheduling of real-life assembly job shop with timed colored petri net, In *Proc. on 7th International Conference on Service Systems and Service Management (ICSSSM)*, 2010, pp. 1–5.

[39] C.Y. Zhang, P. Li, Y. Rao and Z. Guan. A very fast ts/sa algorithm for the job shop scheduling problem, *Computers & Operations Research*, Vol. 35, Issue 1, 2008, pp. 282–294.

[40] G. Zhang, L. Gao and Y. Shi. An effective genetic algorithm for the flexible job-shop scheduling problem, *Expert Systems with Applications*, Vol. 38, Issue 4, 2011, pp. 3563–3573.

[41] Q. Zhang, H. Manier and M.-A. Manier. A modified shifting bottleneck heuristic and disjunctive graph for job shop scheduling problems with transportation constraints, *International Journal of Production Research*, Vol. 52, Issue 4, 2014, pp. 985–1002.

[42] Y. Zheng, L. Lian, Z. Fu and K. Mesghouni. Evolutional Algorithm in Solving Flexible Job Shop Scheduling Problem with Uncertainties, In *LISS 2013*, Springer Berlin Heidelberg, 2015, pp. 1009–1015.