Multi-agent systems in production planning and control: An application to the scheduling of mixed-model assembly lines

Maria Caridi*, Andrea Sianesi

Politecnico di Milano, Dpt. di Economia e Produzione, Piazza Leonardo da Vinci, 32-20133 Milano, Italy

Received 6 April 1998; accepted 11 June 1999

Abstract

This work deals with production smoothing, one of the keys of success of Just In Time and Lean Production. By levelling the load of the workstations, production smoothing allows a regular material flow, shorter manufacturing lead times, and lower work in process. Different solutions to the mixed-model assembly lines sequencing problem have been proposed in literature. In this paper, a Multi-Agent System is presented, which solves this problem according to the theory of autonomous agents. The experimental results show that this innovative approach has a good performance if compared with the traditional ones.

Keywords: Production smoothing; Multi-agent system

1. Introduction

Short-term production planning techniques have known a long evolution starting from the Seventies. After a first era of optimisation dominion, heuristic approach tried to overcome its limitations, yet introducing others. Later, researchers’ attention shifted to other innovative paradigms, which are more effective in modern dynamic and complex contexts. In particular, one of these approaches is Artificial Intelligence (AI): nowadays autonomous agent theory, a product of AI, is one of the most interesting fields of research as far as production planning and control are concerned. The topic of this paper is an application of autonomous agent theory to a particular short-term production planning problem, sequencing of mixed-model lines, which has been studied for years in literature.

After a brief overview of the evolution of short-term production planning techniques across last decades (Section 2), an introduction to autonomous agent theory and multi-agent architectures is reported in Section 3. Later, in Section 4, the problem of sequencing of mixed-model assembly lines is presented, together with the model usually adopted in literature in order to solve this problem. Once modelled, the problem can be solved through different techniques: in particular, the section presents the heuristic and the optimisation approaches, as they have been developed in literature; moreover,
Table 1

<table>
<thead>
<tr>
<th>Era</th>
<th>Control</th>
<th>Approach</th>
<th>Technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimisation</td>
<td>Hierarchical</td>
<td>Automatic</td>
<td>Optimisation or heuristic</td>
</tr>
<tr>
<td>Heuristic</td>
<td>Hierarchical</td>
<td>Automatic</td>
<td>Heuristic</td>
</tr>
<tr>
<td>Complexity</td>
<td></td>
<td>Automatic</td>
<td></td>
</tr>
<tr>
<td>Artificial intelligence</td>
<td>Hierarchical</td>
<td>Automatic</td>
<td>Heuristic</td>
</tr>
<tr>
<td>Neural networks</td>
<td>Hierarchical</td>
<td>Automatic</td>
<td>Heuristic</td>
</tr>
<tr>
<td>Genetic algorithms</td>
<td>Hierarchical</td>
<td>Automatic</td>
<td>Heuristic</td>
</tr>
<tr>
<td>Autonomous agents</td>
<td>Distributed</td>
<td>Automatic</td>
<td>Heuristic</td>
</tr>
<tr>
<td>Interactive schedulers</td>
<td>Distributed (control is delegated to the operator)</td>
<td>Interactive</td>
<td>Heuristic + operator</td>
</tr>
</tbody>
</table>

it proposes a multi-agent architecture which has been recently developed by the authors. The performance of this architecture has been tested and compared with other performing models; the experimental results, shown in Section 5, suggest the existence of significant margins of improvement on the performance of traditional approaches.

2. The evolution of short-term production planning techniques

Four principal moments (eras) of evolution can be identified for short-term production planning techniques:

- optimisation era,
- heuristic era,
- complexity era,
- interactive schedulers era.

The synoptic table of the four above-mentioned techniques is reported in Table 1 and a brief description of each of these is provided subsequently (see also [1]).

2.1. Optimisation era

Seventies and Eighties are the ages of Computer Integrated Manufacturing (CIM). Its characteristic is a strongly hierarchical top-down control and full automation of manufacturing system. The application of this technique has not encountered great success in real world applications mainly because of the following reasons:

- reaching full automation in a manufacturing system takes a long time while, on the other hand, market needs quick response (shorter and shorter time-to-order and time-to-market are required);
- standardisation, which is a natural consequence of automation, makes the firm too strict to face the requirement of product differentiation;
- representing complex real manufacturing systems through computerised algorithms is very hard because it implies the translation of system processes and decisional rules into an analytical model (which contains equations and linear/not-linear relationships among the variables of the system);
- a deep dichotomy exists between algorithmic mono-objective approach and actual multi-objective context.

As soon as these limitations were understood, integrated optimisation started to be considered an utopian paradigm while an opposite approach, the heuristic one, took place.

2.2. Heuristic era

Heuristic paradigm dominated the Eighties. It originated from the need for overcoming the difficulties of a deep modelling of reality, offering a more efficient (quicker) decision support instrument; in fact, generally speaking, an heuristic algorithm is usually nothing but the flow chart of the mental steps followed by the planner in making decisions. Compared with the optimisation
approach, the benefits of such a method are mainly the following ones:

- the logical model is nearer to the physical one,
- the flow chart is built on the basis of the planner’s experience; therefore it is implicitly multi-objective.

The main limitation of this method is it being static (firm rules and priorities are frozen at the moment of the scheduling system project) in contrast to the rapid evolution of the context in which the method is used, in term of products, processes, market requirements, firm strategy, etc. On the other hand, it seems quite impossible to codify the way the system reacts to all the possible failures, lateness or other unforeseen events. This limitation is characteristic of both the optimisation approach and the heuristic one, so that they can be classified as “lowly dynamic techniques” in comparison to the two eras which will be described in the following, which are “highly dynamic techniques”.

2.3. Artificial Intelligence era

Artificial Intelligence era, also known as virtual manufacturing era, extends from the second half of the Eighties to the present and it seems to be the natural (and yet complex) answer to the need of interpreting complex manufacturing systems; this era has given birth to different techniques:

- expert systems: planner’s knowledge is translated into a set of rules of behaviour, stored in the system; each of these is used when certain events happen and they are continually brought up-to-date by the system (learning process); the complexity of the software and the difficulty of knowledge codifying make few the applications of these research-stimulating systems;
- neural networks: they are particular expert systems where knowledge is translated into a pseudo-cerebral architecture; although they simplify the learning process, these systems show the same limitations as the common expert systems;
- genetic algorithms: they are a particular class of adaptive heuristics, based on the principles of biological systems’ evolution (the sole ones surviving in the environment are those embodying particular genetic characteristics); this kind of system shows the same qualitative limitations as neural networks;
- autonomous agent architectures: in the last years manufacturing research started to study possible applications of autonomous agent theory to production planning and control; according to this theory, the decision making process is distributed among intelligent and autonomous entities (autonomous agents), which act in order to reach local objectives. The overall objective, in fact, is split into many local ones, whose persecution should warrant that of the global one. Such an approach allows to overcome the problems of complexity (i.e. large volume of data, production capacity distributed among resources having conflicting objectives, etc.) and uncertainty (i.e. noises coming from the field, uncertainty in lead times, etc.) for the following reasons:
  - co-ordination in the manufacturing system is obtained through the intelligence of the agents;
  - large volume of data is distributed among the agents;
  - the system does not need to manage a complex overall objective function;
  - disturbances can be sorted out by local negotiation among the agents.

Note that decomposition methods in complex optimisation are well known (e.g. the Dantzig and Wolfe [2] algorithm or, more specifically for multi-level POM systems, Graves [3]). Notwithstanding, in comparison with lagrangean decomposition, the autonomous agents approach is particularly interesting in that it provides solutions which draw from distributed information resources, so that it can more effectively react to the uncertainty of the real manufacturing systems thanks to a lean and local decision making.

As far as the limitations of this technique are concerned, the applications of autonomous agents architecture are few at the present; consequently its effectiveness (in pursuing the

---

1 Autonomous agents architectures will be presented in much deeper details in Section 3.
global optimum) and its efficiency (in managing the complexity due to the co-ordination and negotiation among agents) are still hard to evaluate. However, the research on this fascinating field is now in progress. Among the most interesting contributions found in literature, Baker’s research is worth recaptulating. In [4], the author presents a planning and control system designed for small-batch and job-shop manufacturers, which is based on an agent-based fully distributed architecture operating over a network of computers. The implementation of this system shows an improvement in system’s performances, i.e. reduction of work-in-progress inventory, of average time-in-progress, of production costs and tardiness. Other examples of industrial applications of autonomous agent theory can be found in [5].

2.4. Interactive schedulers era

Nineties are the ages of lean/agile/versatile manufacturing. The limitations of CIM paradigm, on the one hand, and of the heuristic approach, on the other, have spurred the search of simplicity in short-term production planning system. This is the era of interactive schedulers: they are the simplest kind of scheduling systems because the plan is not made by a machine but by the planner himself, while the system checks for the feasibility of the decision maker’s choices. Automation usually survives in these systems only in the formulation of a first rough plan (that might be produced with heuristic or optimisation algorithms embedded in the scheduler). Nowadays schedulers are the most commonly used instruments in real world applications.

Fig. 1 classifies the above-presented techniques according to their dynamism (i.e. the possibility of adapting technique application to the environment evolution) and their trade-off between effectiveness and efficiency.

3. Intelligent autonomous agents

In this section, after a brief introduction to the concept of autonomous agent (Section 3.1), a definition for Multi-Agent System (Section 3.2) and for intelligent negotiation is presented (Section 3.3).^2

3.1. What is an autonomous agent?

Research in agents theory has been producing different definitions for the term “agent”. According to Nwana [7], “we have as much chance of agreeing on a consensus definition for the word ‘agent’ as AI researchers have of arriving at one for ‘artificial intelligence’ itself – nil!”. Many collections of agent definition have been claimed until now;^3 the

---

^2See also [6] for an extensive introduction to autonomous agent theory.

^3Refer, for example, [8–11].
following definitions try to summarise the basic concepts shared by many researchers:

- in its broadest meaning, an agent is a system, a set of elements which have a particular relation among themselves and with the environment. Each agent performs a particular function for other agents and this function gives it resources to survive;
- the meaning of the term “autonomy” comes from etymology autos (itself) and nomos (rule or law): autonomous agents are then independent systems developing by themselves the laws and strategies according to which they regulate their behaviour.
- lastly, intelligence is an autonomous agent property which allows the agent to interpret reality building a representation of it. The ability to handle representations in order to adapt itself quickly to the environment dynamics is the most distinguishing characteristic of an intelligent system.

The three above-defined terms explain the general concept of intelligent autonomous agent. In particular, this definition is equally applicable to software agents; as Green et al. [10] observe, in computational context the term “agent” has its background in the early work on Artificial Intelligence (AI) when researchers concentrated on trying to create artificial entities which mimicked human abilities. Software agents loosely conform to the above definition for “intelligent autonomous agent” and can basically be described as inhabiting computer and networks, assisting users with computer-based tasks [7]. Unlike most software which must be specifically called upon to act, this so-called software agent continuously acts on its own initiative; moreover, the agent system is designed so that when an agent is meeting its goals, it is really meeting the goals it has been programmed to meet by its designer [13]. Researchers and software companies have set high hopes on software agents, which “know” users’ interests and act autonomously on their behalf: instead of exercising complete control, people will be engaged in a co-operative process in which both human and computer agents initiate communications, monitor events and perform tasks to meet users’ goal [10].

3.2. What is a Multi-Agent System (MAS)?

According to O’Hare and Jennings’ [14] definition, a MAS is a network of problem solvers that work together to solve problems that are beyond their individual capabilities. A constraint satisfaction problem is so subcontracted to different problem solving agents with their own interests and goals.

The increasing interest in MAS research is due to the significant advantages inherent in such systems, including their ability to:

- solve problems that may be too large for a centralised single agent due to resource limitations or the sheer risk of having one centralised system;
- allow for the interconnecting and interoperation of multiple existing legacy systems, e.g. expert systems, decision support systems, etc.;
- provide solutions to inherently distributed problems, e.g., air traffic control [15];
- provide solutions which draw from distributed information sources;
- provide solutions where the expertise is distributed, e.g. in health care provisioning;
- offer conceptual clarity and simplicity of design;
- tolerate uncertain data knowledge.\(^5\)

MAS advantages may not seem to be as innovative as it has been stated before. It is well known, in fact, that system separation and subsequent co-ordination, and moreover, discrete information and private preferences have already been utilised in POM, OR and particularly in the management accounting literature (just think of transfer pricing!). Although, the innovation of this approach has

\(^4\)Hewitt [12].

\(^5\)Green et al. [10].
been found neither in the organisation structure of the agent-based system (centralised, hierarchical, eterarchical, etc.) nor in the way they communicate (blackboard, message passing, etc.): all organisations or communication patterns have already been explored in specialised literature. The innovation of MAS approach is its attempt of applying these well known patterns to a context where the actors are not humans or software objects but entities which are autonomous and intelligent, acting according to rules they modify by themselves, so that global solution draws from the information they have locally gathered, from their beliefs, motivations and consequent actions.

3.3. The concept of negotiation

A basic definition for negotiation is that of Bus-ssmann and Muller [16]: negotiation is the commun-ication process of a group of agents in order to reach a mutually accepted agreement on some matter. For example, in a manufacturing context the agreement might be about price, about start and finish time of operations, about service/quality performance and so on. Hence, the basic idea behind negotiation is reaching a consensus.

Negotiation can be competitive or co-operative depending on the co-operative behaviour of the individual agents involved. Referring to Green et al. [10] for further details,6 in this section only co-operative negotiation will be presented, since the architecture proposed in this paper uses this kind of mechanism for co-ordination among agents.

As for co-operative negotiation, the strategy described below can be applied to a MAS where the individual agents are co-operative and will collaborate in order to achieve a common goal, for the best interest of the system as a whole.

The general strategy is that negotiation begins with one, some or every agent making a proposal. The other agents evaluate and check the proposals and the negotiation cycle resumes with a new proposal or proposals in the light of the newly gleaned information.

Research in manufacturing applications of MAS to date has developed architectures in which the decision of “which part has to be worked by which machine” is made through a message passing among agents and a bidding protocol to achieve mutual agreement. In general, each part and each resource (machine, tool, fixture, transport devices and so on) of the system is supposed to be provided with an intelligent agent and all decisions are made through negotiation. A metaphor will help to de-scribe this mechanism.

Negotiation resembles stock exchange in an open marketplace: investors (part agents) enter the system with some fictitious currency and try to achieve their profit (set of part’s objectives) by buying listed stocks (resource agents).

System performance results from the equilibrium of the price-offer law: the greater the demand for a machine, the greater will be its selling price: each part shall evaluate the convenience of buying a powerful and expensive machine or a less per-forming and yet cheaper machine. As a result, the system will adjust itself smoothly.

Notice that this is a multiple-way and multiple-step negotiation [17]. It is multiple-way since a part usually needs the simultaneous service of several agents (machine, tool, fixture, transport devices and so on) in order to make each operation of its cycle: therefore, a part agent will negotiate with different resources at a time. It is also multiple-step since both parts and resources have opportunities to deny or accept an offer during the negotiation concerns: final agreement is made only when all parts and resources commit themselves through multiple-step negoti-ation.7

---

6 Briefly, competitive negotiation involves independent agents, with independent goals that interact each other, so they are not a priori co-operative, i.e., they are competitive; on the other hand, co-operative negotiation involves agents to pursue a single global goal, so that they act collaboratively.

7 A detailed example of multiple-way, multiple-step negotiation for job-shop scheduling of products with multi-level BOM is available in [18].
4. The mixed-model lines scheduling problem: Different solving approaches

This section presents the problem of products sequencing on a mixed-model assembly line and various approaches for its solution, encompassing the autonomous agents architecture which has been studied by the authors for this kind of problem.

4.1. The problem and its mathematical representation

A four-level manufacturing system is the subject of the model; the levels are numbered as follows:

- level 1 – products,
- level 2 – sub-assemblies,
- level 3 – components,
- level 4 – raw materials.

Each product (level 1) is made up of a variety of sub-assemblies (level 2), which consist of different components (level 3), which in turn are made up of raw materials (level 4).

Given a set of products, various schedules can be followed for assembling them. The aim of the model is determining a balanced schedule, that is a schedule producing each product concurrently. In this way, small batches of each product are produced often, so that the production mix is synchronised with the market demand; as a consequence, inventories are kept small. Such an assembly line is called mixed-model line.

The following variables are introduced:

\[ j \] is the level number; \( j = 1, \ldots, 4, \]

\[ n_j \] is the number of outputs at level \( j; \) \( j = 1, \ldots, 4, \]

\[ d_{i,1} \] is the demand for product \( i; \) \( i = 1, \ldots, n_1, \]

\[ t_{i,j,l} \] is the unit of output \( i \) at level \( j \) used to produce one unit of product \( l; \) \( i = 1, \ldots, n_j; \]

\( j = 1, \ldots, 4; \) \( l = 1, \ldots, n_1 \) (for \( j = 1, \quad t_{i,1,l} = 1 \) if \( i = l \) and 0 otherwise).

From \( d_{i,1} \) and \( t_{i,j,l} \) the following quantities can be determined:

\[ d_{i,j} = \sum_{h=1}^{n_l} t_{i,j,h} \cdot d_{h,1} \] demand for output \( i \) at level \( j \); \( i = 1, \ldots, n_j; \]

\( j = 2, \ldots, 4, \)

\[ DT_j = \sum_{i=1}^{n_j} d_{i,j} \] total demand for production at level \( j; \) \( j = 1, \ldots, 4, \)

\[ r_{i,j} = (d_{i,j})/(DT_j) \] output \( i \) of level \( j \) for each product entering the line at level 1; \( i = 1, \ldots, n_j; \)

\( j = 1, \ldots, 4, \)

\( DT_1 \) products must be assembled on the final assembly line during the planning horizon, that is there are \( DT_1 \) consecutive stages; each product has to be assigned to a stage. A schedule is an assignment of \( DT_1 \) products to the line. The schedule is represented in the model by the variables \( x_{i,j,k} \), which are the units of product \( i \) produced during stages \( 1, \ldots, k \). The production at the lower levels can be determined as follows:

\[ x_{i,j,k} = \sum_{h=1}^{n_l} t_{i,j,h} \cdot x_{h,1,k} \] units of output \( i \) at level \( j \) produced during stages \( 1, \ldots, k \); \( i = 1, \ldots, n_j; \)

\( j = 2, \ldots, 4 \).

If production were perfectly levelled, after \( k \) stages the total output \( x_{i,j,k} \) (actual output) of part \( i \) at level \( j \) should be \( k \cdot r_{i,j} \) (theoretical output); equality is not always possible because \( k \cdot r_{i,j} \) is not an integer. The aim of the model is finding a schedule that minimises the difference between actual output and theoretical output of every part at every level. The following expression interprets this objective:

\[ \min \sum_{k=1}^{DT_1} \sum_{j=1}^{4} \sum_{i=1}^{n_j} w_j (x_{i,j,k} - k \cdot r_{i,j})^2 \]

where the weights \( w_j \) express the relative importance of deviation at the various levels.

Equipped with these definitions and notations, the sequencing model is formulated as follows:

\[ \min \sum_{k=1}^{DT_1} SDQ_k = \sum_{k=1}^{DT_1} \sum_{i=1}^{n_j} w_j (x_{i,j,k} - k \cdot r_{i,j})^2, \]

\[ x_{i,j,k} = \sum_{h=1}^{n_l} t_{i,j,h} \cdot x_{h,1,k} \] \( \forall i = 1, \ldots, n_1, \)

\( \forall j = 1, \ldots, 4, \)

\( \forall k = 1, \ldots, DT_1, \)

\[ x_{i,1,DT_1} = d_{i,1} \] \( \forall i = 1, \ldots, n_1, \)

\[ \sum_{i=1}^{n_1} x_{i,1,k} = k \] \( \forall k = 1, \ldots, DT_1, \)

\( 0 \leq x_{i,1,k} - x_{i,1,(k-1)} \leq 1 \) \( \forall i = 1, \ldots, n_1, \)

\( \forall k = 1, \ldots, DT_1, \)

\( x_{i,1,k} \geq 0 \) \( \forall i = 1, \ldots, n_1, \)

\( \forall k = 1, \ldots, DT_1. \)
4.2. Heuristic approach

Finding the optimum of an integer not-linear problem such as (1) is hard; for this reason, starting from the Eighties manufacturing research fell back on the heuristic approach.\textsuperscript{9}

Monden [21] gave research a very interesting contribution: his \textit{Goal Chasing Method} solved efficiently problem (1) when \( j = 1, 2 \), that is to say when products to be sequenced have a two-level bill of materials (two-level BOM). This algorithm was used with success in Toyota to schedule automobile final assembly lines.

Later, Miltenburg and Sinnamon [19] proposed an extension of Monden’s [21] method, called Extended Goal Chasing Method (EGCM); this algorithm extended the previous one to a multi-level BOM system. Since this method poses itself at the basis of the architecture which will be presented in Section 4.4, it is worth explaining in further detail.

EGCM is a greedy algorithm according to which problem (1) is divided into DT \(_1\) sequential and independent steps. For each step \( k \), the scheduled product \( s \) satisfies the following expression:

\[
\sum_{j=1}^{4} \sum_{i=1}^{n_j} w_j[(x_{i,j,k-1} + t_{i,j,k}) - k \cdot r_{i,j}]^2
\]

\[
= \min_{1 \leq p < n_j} \sum_{j=1}^{4} \sum_{i=1}^{n_j} w_j[(x_{i,j,k-1} + t_{i,j,p}) - k \cdot r_{i,j}]^2.
\]

The algorithm iterates for \( k = 1, \ldots, DT \_1 \).

4.3. Optimisation approach

Recent research has proposed a new interesting algorithm that can potentially find the optimum of (1) for a two-level BOM system. This work, by Bautista et al. [22], tries to overcome the myopic behaviour of Monden’s GCM or other improving algorithms\textsuperscript{10} through a powerful heuristic based on the principles of Bounded Dynamic Programming (BDP), which combines features of Dynamic Programming (i.e. determination of the minimum path in a graph) with features of Branch and Bound (i.e. research of a bound for the optimal solution of the problem).\textsuperscript{11}

Problem (1) is presented as a minimum path problem, through a connected graph without loops and with DT \(_1\) levels (see Fig. 2).

With each vertex \( v \) at level \( k \), a vector \( X_{k,v} \) of components \( x_{i,k} \) is associated such that

\[
\sum_{i=1}^{n_k} x_{i,k} = k, \quad 0 \leq x_{i,k} \leq d_{i,1}, \quad i = 1, \ldots, n_1.
\]

Obviously, at level 0 and at level DT \(_1\) there is a single vertex corresponding to

\[
X_0 = [0 \ 0 \ \cdots \ 0]
\]

and

\[
X_{DT_1} = [d_{1,1} \ d_{2,1} \ \cdots \ d_{n_1,1}].
\]

Between a vertex \( v_1 \) of level \( k \) and a vertex \( v_2 \) of level \( k + 1 \) there is an arc if the vertices associated with them, namely \( X_{k,v_1} \) and \( X_{k+1,v_2} \), satisfy the following condition:

\[
X_{k+1,v_2} - X_{k,v_1} = I_h \quad \text{for some } h = 1, \ldots, n_1;
\]

where \( I_h \) is the \( h \)-th column of the \((n_1 \times n_1)\) identity matrix.

Moreover, a weight is associated with the vertex \( v_2 \), which expresses the gap existing between

\textsuperscript{9}See Kubiak [20] for a survey about the most significant heuristic approaches developed in literature for mixed-model lines balancing problem.

\textsuperscript{10}Among the others, see in particular Bautista et al. [22] for a presentation of various heuristic extensions of GCM for a two-level assembly line.

\textsuperscript{11}The principles of Bounded Dynamic Programming in a general form are described by Bautista et al. [23].
theoretic and effective output at step $k + 1$, given the sequence of products up to level $k$ and given the fact that, at level $k + 1$, product $h$ is added to the sequence.12

Any path from $X_0$ to $X_{DT_1}$ is a feasible solution of Problem (1) and its weight expresses the value of the objective function. Finding the optimal solution of Problem (1) is equivalent to finding the shortest path from $X_0$ and $X_{DT_1}$ in the graph.

In spite of this, considering the high number of vertexes of the graph (exponential in $n_1$), an optimal procedure (such as that of pure dynamic programming) is prohibitive. Then, Bautista et al. [22] proposed an algorithm which explores the graph, level by level, according to dynamic programming approach, yet it does not evaluate all the vertexes for each level. In fact, the procedure calculates an upper bound for the minimum path and, with each vertex, it associates a lower bound of the weight of the most convenient path passing through it. By comparing the two bounds (namely, if the lower bound is higher than the upper bound), a vertex can be eliminated from the graph. Moreover, the maximum number of vertices explored in each step of the procedure is fixed a priori as a consequence, it can happen that a vertex belonging to the optimal path is eliminated a priori.

Since this procedure does not necessarily find the optimal solution, it is an heuristic. Yet, at the end of the procedure it is possible to check whether a certain condition is verified: if it is, the solution obtained is surely optimal.

Bautista et al. [22] prove that this procedure, yet basically heuristic (but potentially optimal), performs well when compared to others proposed in literature for two-level BOM systems. An extension to multi-level BOM systems has been proposed in [24].

4.4. The autonomous agent architecture

In this paragraph a resolution of Problem (1) based on autonomous agents’ theory is proposed. This architecture shares with EGCM the idea of dividing Problem (1) into independent and sequential steps.

Yet, two main differences between the two approaches have to be cleared out:

- the decision of which product is to be scheduled at each step of the algorithm is not made by comparing the $n_1$ values of expression (2) but through a negotiation among independent autonomous agents
- moreover, the $n_1$ compared values are not calculated on the basis of a global information (as in expression (2)) but they are determined by each finished product by itself, exclusively on the basis of its local information (i.e. the information related to its bill of materials).

The second above-highlighted difference interprets the MAS characteristic of providing solutions which draw from distributed information resources. There is no need of gathering information in a unique centralised database, in fact different information points exist in the system and each of them controls a portion of information (note that information portions can be overlapped, i.e. the same kind of component can be used in more than one finished product).

The latter characteristic is a sufficient condition for considering the model presented in this section a MAS, although each entity is a weak agent (see [25]) in that it does not modify its rules during its life.

It will be interesting to observe that such a local-information based system can obtain better performances than the global-information based EGCM.

The problem is represented through a multi-agent system where the number of agents is equal to the global number of products, sub-assemblies, components and raw materials plus one, associated with the mixed-model assembly line. Globally, there are

$$1 + \sum_{j=1}^{4} n_j$$

agents.

As for EGCM, the proposed mechanism tries to determine, at each step $k$ ($k = 1, \ldots , DT_1$) of the problem, which product minimises the difference.
between theoretical and actual output for all the products, sub-assemblies, components and raw materials of the system.

As it will be clearer later, the co-ordination among agents is obtained through a multiple-way, one-step negotiation (see Section 3.3).

At each step \( k (k = 1, \ldots, DT_1) \), the assembly line decides which product to schedule, aiming to level production and part consumption. In order to do that, it sends a bid requirement to each product, that is each product has to determine the price it will pay the line for being scheduled in the current step. This price expresses the value of the schedule for the product per se and for all the parts of its BOM (since they are consumed if the product is scheduled). Once the line has received all the bids, it chooses the most convenient one (the highest) and it schedules the corresponding product.

The analytical expression of the local price of each product has been studied in order to take into account the overall objective of levelling the consumption of the parts at each level of its BOM. In particular, the price of a product is the sum of the prices of all the parts at every level of its BOM. In order to determine the price of the generic part \( i \) at level \( j \), the following expressions are introduced:

\[
\Delta_{i,j,k} = k \cdot r_{i,j} - x_{i,j,k},
\]

\[
m_{i,j,h} = \frac{t_{i,j,h} \cdot d_{h,(j-1)}}{d_{i,j}}
\]

where \( \Delta_{i,j,k} \) is the gap between theoretical and actual output of part \( i \) at level \( j \) at step \( k \), while \( m_{i,j,h} \) is the ratio of demand of part \( i \) at level \( j \) consumed by its parent \( h \). The price of part \( i \) at level \( j \) is given by the following recursive equation:

\[
V_{i,j,k,h} = \left( \Delta_{i,j,(k-1)} + \sum_{z=1}^{n_{i,j}} V_{z,(j+z),(k,i)} \right) m_{i,j,h} \tag{3}
\]

Note that the first term in parentheses expresses the value of the schedule for part \( i \), while the second one contains the value for all its sons in BOM. The term \( m_{i,j,h} \) increases the price in case of large consumption of part \( i \) by its parent \( h \): in fact, the higher its consumption, the higher the probability of reducing the part’s gap.

The procedure is very simple: at each step \( k (k = 1, \ldots, DT_1) \), the assembly line asks each product \( i \) to construct a bid, that is to determine its price \( V_{i,1,k,L} \) (\( L \) indicates the assembly line). Referring to (3), the first addendum \( \Delta_{i,1,(k-1)} \) can be determined by the product itself, while the determination of the second addendum requires that product \( i \), in turn, asks its sons to construct a bid. This is a top-down bid-call mechanism: starting from the products, each level \( j \) receives calls from

Fig. 3. Example of top-down bottom-up negotiation mechanism.
Table 2
Initial data

<table>
<thead>
<tr>
<th>Part</th>
<th>$A_{i,j,k}$</th>
<th>$m_{i,j,k}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product 1 (P1)</td>
<td>10</td>
<td>—</td>
</tr>
<tr>
<td>Sub-assembly 1 (SA1)</td>
<td>15</td>
<td>0.1</td>
</tr>
<tr>
<td>Sub-assembly 2 (SA2)</td>
<td>8</td>
<td>0.2</td>
</tr>
<tr>
<td>Component 1 (C1)</td>
<td>12</td>
<td>0.7</td>
</tr>
<tr>
<td>Component 2 (C2)</td>
<td>30</td>
<td>0.5</td>
</tr>
<tr>
<td>Raw material 1 (RM1)</td>
<td>60</td>
<td>0.4</td>
</tr>
<tr>
<td>Raw material 2 (RM2)</td>
<td>100</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 3
Top-down bid call

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>SA1</td>
<td>Prepare your bid for my scheduling</td>
</tr>
<tr>
<td>P1</td>
<td>SA2</td>
<td>Prepare your bid for my scheduling</td>
</tr>
<tr>
<td>SA2</td>
<td>C1</td>
<td>Prepare your bid for my consumption</td>
</tr>
<tr>
<td>SA2</td>
<td>C2</td>
<td>Prepare your bid for my consumption</td>
</tr>
<tr>
<td>C2</td>
<td>RM1</td>
<td>Prepare your bid for my consumption</td>
</tr>
<tr>
<td>C2</td>
<td>RM2</td>
<td>Prepare your bid for my consumption</td>
</tr>
</tbody>
</table>

Table 4
Bottom-up bid collection

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>$V_{i,j,k,h}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>RM1</td>
<td>C2</td>
<td>$60 \times 0.4 = 24$</td>
</tr>
<tr>
<td>RM2</td>
<td>C2</td>
<td>$100 \times 0.1 = 10$</td>
</tr>
<tr>
<td>C2</td>
<td>SA2</td>
<td>$(30 + 24 + 10) \times 0.5 = 32$</td>
</tr>
<tr>
<td>C1</td>
<td>SA2</td>
<td>$12 \times 0.7 = 8.4$</td>
</tr>
<tr>
<td>SA2</td>
<td>P1</td>
<td>$(8 + 32 + 8.4) \times 0.2 = 9.68$</td>
</tr>
<tr>
<td>SA1</td>
<td>P1</td>
<td>$15 \times 0.1 = 1.5$</td>
</tr>
<tr>
<td>P1</td>
<td>Assembly line</td>
<td>$10 + 9.68 + 1.5 = 21.18$</td>
</tr>
</tbody>
</table>

5. Experimental results

The agent based model has been implemented using KQML (Knowledge Query and Manipulation Language),\(^{13}\) as a communication language among agents, in a LALO (Langage d'Agents Logiciel Objet)\(^{14}\) programming environment.

The experiments have been carried out under the following hypotheses:

- 6 kinds of products to be assembled;
- demand:
  - 2 kinds of demand distribution among the products (D1: homogeneous D2: concentrated on 50% of the products);
  - 6 vectors of demand for each kind of demand distribution;
- complexity:
  - 5 kinds of BOMs determined by the triple: $n_1/n_2/n_3$;
- part sharing:
  - 2 cases in terms of number of parents for each sub-assembly (“1 : 2”: each sub-assembly is son of 50% of the products; “1 : all”: each sub-assembly is son of each product);
  - 2 cases in terms of number of parents for each component (“1 : 2”: each component is son of 50% of the sub-assemblies; “1 : all”: each component is son of each sub-assembly).

In order to test the performance of the model, its resulting objective function value has been

\(^{13}\) See [26].
\(^{14}\) See [27].
compared with that obtained by the Bautista et al. [22] algorithm and other performing ones. In particular, in each experiment the following parameter has been measured for each tested algorithm $X$:

$$\Delta SDQ\% = \frac{SDQ_X - SDQ_{\text{Bautista et al.}}}{SDQ_{\text{Bautista et al.}}} \times 100.$$  

Considered the aim of the experiments, the weights $w_j$ have been set at 1 in order to filter their influence on the performance of the tested algorithms.

Observing the experimental results, the agent model shows a good effectiveness in comparison with the other algorithms in particular:

- when demand is homogeneous, the performance of the agent model is better than that of the other algorithms, except for Bautista et al. [22];
- when demand is not-homogenous, it results that the more complex the BOM, the worse the performance of the model; in these cases its average performance is third after Bautista et al. [22] and Monden [21].

The following tables report the experimental results (in terms of average $\Delta SDQ\%$) for the case of part sharing $1:2/1:2$.

Tables 5 and 6 show that the performance of the agent-based architecture varies from 17.5% to 197.1% in comparison with Bautista et al. [22], depending on the distribution of demand and on the complexity of BOM. This result is not surprising, since the Bautista et al. [22] algorithm combines Dynamic Programming and Branch and Bound: thanks to this approach the choice taken in each step of the algorithm can take into consideration the impact on the global optimum, preventing from greedy solutions. On the contrary, the multi-agent approach is myopic by definition (each agent performs its function aiming at its local objective).

On the other hand, it is worth highlighting that the MAS approach reports better performances than the other myopic approaches. The reason is based on the way each single agent defines its price: the usage of the coefficient $m$ allows to strengthen the local choices that can potentially improve the global performance.

Moreover, the experiments show that:

- since the price (3) is determined as the sum of not-squared terms in parentheses, high positive values can be balanced by high negative values;\(^{16}\) as a consequence, there is not a fair correspondence between the price set by a product and the increment of the SDQ due to its production (in fact SDQ is a sum squared terms);

\(^{15}\)The other algorithms compared are: [19,21], “1-step” and “2-step” algorithms in [28,29].

\(^{16}\)Note that the gap between theoretical and actual output can assume both positive and negative value.
• in many cases, the resulting schedule has a particular characteristic: after a variable number of adjustment stages, sequence becomes repetitive (namely, the sequence can be divided in a certain number of identical sub-sequences).

These two elements have spurred the enrichment of the model in this way: once all the products have been scheduled, the assembly line analyses the effect of its choice on the sequence and on the global SDQ, in order to decide whether its decision has to be revised. If it does, the assembly line asks products to co-operate in order to find another solution for the current step, which can improve the global result.

Rough behaviour rules of co-operation have been formulated and implemented in a narrow set of experiments. The results show that this enrichment of the model improves the performance of the system, since co-operation makes the agent model less myopic: co-operation is in fact a learning process (it results from an analysis a posteriori of the scheduled sequence) through which, on the basis of the knowledge acquired, the system (namely, the assembly line agent) tries to improve the global objective.

6. Conclusions

This paper has presented three different approaches to short-term production planning problem of mixed-model lines.

As stated in the previous sections, the optimisation approach (i.e. [22]) is more suitable for static contexts, where the effort to determine the optimal solution (mainly due to algorithmic complexity, long processing time, etc.) is justified.

On the other hand, heuristic algorithms are much more simple, in terms of complexity, and they require shortest processing time: so that they are more suitable for dynamic contexts, where frequent re-programming is needed.

Last but not least, the autonomous agents approach is a different and innovative technique. The classical problem of short-term production planning of mixed-model lines is basically mono-objective and deterministic, and so the utilisation of innovative resolution techniques seem not justified (i.e. the agent-based approach does not improve the results of the Bautista et al. [22] algorithm). Nevertheless, the autonomous agents architecture which has been presented in this paper is interesting at least for the following two reasons:

• the proposed architecture is characterised by a distributed control (various autonomous entities fill the system and each of them is endowed with intelligence and objectives); as a consequence, such a model is modular, that is new bonds or new degrees of freedom (for example, the possibility of working the same product on more than one assembly line indifferently) or new objectives (for example, the minimisation of setup time) can be easily introduced without heavy modifications of the architecture: for example, they can be faced with a redefinition of agents’ objective functions or with the addition of new agents to the system

• the agent architecture shows a good performance in comparison with traditional algorithms and it can potentially find a better solution than that obtained by other heuristics, thanks to the introduction of co-operation, a learning process which reduces the myopia of this procedure in comparison with others.

The state-of-art of research in autonomous agent theory has not yet proposed efficient and stable instruments for software development to make possible the application of this technique to real manufacturing contexts. As a consequence, only a huge experimental activity is possible at the moment, as the one is in progress at Politecnico di Milano. In any case, the experimental results obtained until now show that the autonomous agent technique has a great potential which will be surely exploited in the future through real applications, as soon as the necessary instruments will be developed.

References


[5] H.V.D. Parunak, Applications of distributed artificial intelligence in industry, in: O’Hare, Jennings (Eds.), Foundation of Distributed Artificial Intelligence, Wiley InterScience, New York, 1996 (Chapter 4).


