International Journal of Production Research

ISSN: 0020-7543 (Print) 1366-588X (Online) Journal homepage: https://www.tandfonline.com/loi/tprs20

Alternative graph formulation for solving complex factory-scheduling problems

D. Pacciarelli

To cite this article: D. Pacciarelli (2002) Alternative graph formulation for solving complex factory-scheduling problems, International Journal of Production Research, 40:15, 3641-3653, DOI: 10.1080/00207540210136478

To link to this article: https://doi.org/10.1080/00207540210136478

Published online: 14 Nov 2010.

INT. J. PROD. RES., 2002, VOL. 40, NO. 15, 3641–3653

Alternative graph formulation for solving complex factory-scheduling problems

D. PACCIARELLI†

The adaptation of the academic job shop scheduling model to industrial practice is well known to be rich of difficulties owing to the gap between scheduling theory and practice. On the other hand, several authors observed that the disjunctive graph formulation of Roy and Sussman was more robust than the job shop model itself. In fact, the disjunctive graph can be easily adapted to deal with many practical issues. This paper moves a step closer in the direction of generalizing the existing mathematical models in order to capture relevant aspects of complex factory scheduling problems. It considers an extension of the disjunctive graph, called an alternative graph, and allows it to be modelled in a precise way to solve effectively a number of complex practical scheduling problems for which there were no successful methodologies. Several examples are presented and a complex industrial application arising in the production of steel is provided. Finally, the performance of a new fast heuristic both on real data and several instances from the literature is discussed in relation to the job shop scheduling problem.

1. Introduction

The job shop scheduling problem is the problem of allocating machines to competing jobs over time, subject to the constraint that each machine can handle at most one job at a time. A considerable number of algorithmic improvements have been carried out in the past 30 years to solve the academic version of the job shop problem (e.g. Adams *et al.* 1988, Applegate and Cook 1991, Carlier and Pinson 1994, Nowicki and Smutnicki 1996). Most of these works are based on the disjunctive graph formulation of Roy and Sussman (1964). On the other hand, the adaptation of the academic job shop scheduling model to industrial practice is rich in difficulties owing to a number of additional constraints that make it unsuitable for solving industrial problems. Several authors observed that the disjunctive graph formulation of Roy and Sussman could be easily adapted to deal with many practical issues (e.g. White and Rogers 1990, Schutten 1998). A strong limitation that still remains in these models is that they disregard the behaviour of the jobs between consecutive operations. In other words, with the disjunctive graph formulation, it is necessary to assume that intermediate buffers have infinite capacity and that a job can be stored for an unlimited amount of time. In fact, there are practical situations in which the buffer capacity has to be taken into account, at least at some stages of the production process, and there may be also limits on the amount of time a job can spend in a buffer between two consecutive operations. Examples of such situations arise in many different industrial settings, such as in the production of steel (e.g. Dorn

Revision received November 2001.

† Dipartimento di Informatica e Automazione, Università 'Roma Tre', Via della Vasca Navale, 79, I-00146 Rome, Italy. e-mail: pacciarelli@dia.uniroma3.it

and Shams 1996, Lixin Tang *et al.* 2000), of concrete wares (Grabowski *et al.* 1997) and of chemical products (Reklaitis 1982).

The present paper uses the alternative graph model of Mascis and Pacciarelli (2000) to model the above cited constraints. The alternative graph is based on a generalization of the disjunctive graph of Roy and Sussman. With the alternative graph formulation, it is possible to model in a precise way and solve effectively a number of complex practical scheduling issues for which there were no successful methodologies. In particular, cited are the maximum storage time for a job, or even for a pair of consecutive operations in a job, the buffer capacity for a machine, i.e. the maximum number of jobs that can be stored in the buffer at the same time, the unavailability of the machines in some given time intervals and the specification of non-relaxable time windows for the operations.

The paper is organized as follows. Section 2 gives a review of related works. Section 3 introduces the notation and alternative graph formulation. Then, several practical issues are formulated by means of small examples and a complex scheduling problem arising in the production of steel is reported. Section 4 presents a new fast heuristic procedure for determining a good solution and its performance is discussed. Conclusions follow in Section 5.

2. Literature review

The applications and the literature related to the present work are reviewed briefly. In the literature on machine scheduling, a number of extensions of the traditional job shop model have been studied. For example, White and Rogers (1990) discussed the adaptation of the disjunctive graph formulation of Roy and Sussman (1964) to deal with several practical problem feautures, such as assembly and disassembly sequences, set-ups, due dates, release times, maintenance operations, material-handling delays, and many other operational side constraints. Ovacik and Uzsoy (1992, 1997) and Schutten (1998) extended the shifting bottleneck procedure of Adams *et al.* (1988) to deal with such practical features.

A different research direction focuses on several scheduling problems for which the disjunctive graph model is not suitable. Among these problems are job shop scheduling with limited capacity buffers. Hall and Sriskandarajah (1996) modelled the absence of intermediate buffers as a *blocking* constraint. In this case, a job, having completed processing on a machine, remains on it until the next machine becomes available for processing. McCormick *et al.* (1989) studied a flow shop scheduling problem in an assembly line having finite capacity buffers between machines. They modelled the positions of the intermediate buffers as machines with zero processing time. The problem with limited buffers can therefore be studied as a blocking problem where all machines have no intermediate buffers.

Other relevant cases in which the job shop model with unlimited buffers is not suitable arise in the management of perishable items. A commodity is said to be perishable if some of its characteristics are subject to deterioration with respect to customer/producer requirements. This field of application has been studied mainly in the framework of inventory theory (e.g. Nahmias 1982). In practice, as shown by Arbib *et al.* (1999), product decay often also occurs at some particular stages of the production process, this decay depending on both the product and production step. In these cases, finding adequate scheduling policies is critical to maximize the throughput and reduce losses and costs due to perishing. The perishability issue is approached in various ways in the literature on scheduling, but most of the scheduling models adopted for this kind of productions typically suffer from a lack of information. For instance, an approximation often introduced when scheduling perishable goods with a high decay rate is the introduction of tight no-wait constraints (e.g. Pinedo 1995, Hall and Sriskandarajah 1996, Grabowski *et al.* 1997). A no-wait constraint occurs when two consecutive operations must be performed without any interruption. See Hall and Sriskandarajah (1996) for an extensive survey on the machine scheduling problems with blocking and no-wait in process.

Additional extensions of the job shop problem considered by various authors concern the release and due dates for the jobs, and the availability of the machines. In the latter case, a machine can process jobs only during prespecified time windows. These two extensions can be jointly considered by specifying some non-relaxable time windows for the operations, i.e. earliest/latest possible start time windows. This problem has been extensively studied in the form of a constraint satisfaction problem by several researchers from the field of Artificial Intelligence (e.g. Sadeh *et al.* 1995). Further extensions of the job shop with unlimited buffers can be found, for example, in Ovacik and Uzsoy (1997) and Strusevich (1997).

3. Notation and problem definition

This section introduces the notation and a more precise specification of the problem under consideration. In the usual definition of the job shop problem, a job must be processed on a set of machines. The sequence of machines for each job is prescribed and the processing of a job on a machine is called an *operation* and cannot be interrupted.

This paper focuses on the sequencing of operations rather than jobs. We have therefore a set of operations $N = \{o_0, o_1, \dots, o_n\}$ to be performed on m machines $\{m_1, m_2, \dots, m_m\}$. Each operation o_i requires a specified amount of processing p_i on a specified machine $M(i)$, and cannot be interrupted from its starting time t_i to its completion time $c_i = t_i + p_i$. o_0 and o_n are dummy operations, with zero processing time, that are called 'start' and 'finish' respectively. Each machine can process only one operation at a time.

There is a set of precedence relations among operations. A *precedence relation* (i, j) is a constraint on the starting time of operations o_j, with respect to t_i. More precisely, the starting times of the successor o_j must be greater or equal to the starting time of the predecessor o_i plus a given *delay* f_{ij}, which in our model can be either positive, null or negative. A positive delay may represent, for example, the fact that operation o_j may starts processing only after the completion of o_i, plus a possible set-up time. A delay $\leqslant 0$ represents a synchronization between the starting times of the two operations. Finally, it is assumed that o_0 precedes $o_1, \dots, o_n$, and o_n follows $o_0, \dots, o_{n-1}$.

Precedence relations are divided into two sets: fixed and alternative. Alternative precedence relations are partitioned into pairs. They usually represent the constraints that each machine can process only one operation at a time.

A *schedule* is an assignment of starting times t_0, $t_1, \dots, t_n$ to operations $o_0, o_1, \dots, o_n$ respectively, such that all fixed precedence relations, and exactly one for each pair of the alternative precedence relations, are satisfied. Without loss of generality, it is assumed that $t_0 = 0$. The goal is to minimize the starting time of operation o_n. This problem can be therefore formulated as a particular *disjunctive program*, i.e. a linear program with logical conditions involving operations 'and' ($\wedge$, conjunction) and 'or' ($\vee$, disjunction), as in Balas (1979).

Problem 3.1

$$\begin{array}{lll} \min & t_n & \\ s.t. & t_j - t_i \geq f_{ij} & (i,j) \in F \\ & (t_j - t_i \geq a_{ij}) \vee (t_k - t_h \geq a_{hk}) & ((i,j),(h,k)) \in A. \end{array}$$

Associating a node to each operation, Problem 3.1 can be usefully represented by the triple $\mathcal{G} = (N, F, A)$ that we call *alternative graph*, which is as follows. There is a set of nodes N, a set of directed arcs F and a set of pairs of directed arcs A. Arcs in the set F are *fixed* and f_{ij} is the length of arc $(i,j) \in F$. Arcs in the set A are *alternative*, and a_{ij} is the length of the alternative arc (i,j). If $((i,j),(h,k)) \in A$, we say that (i,j) is *the alternative* of (h,k). In our model, the arc lengths can be either positive, null or negative.

A schedule is obtained by selecting exactly one arc from each alternative pair in such a way that no positive length cycle is generated in the resulting graph. The makespan of the schedule is the length of a longest path from nodes 0 to n in this graph.

The alternative graph is a generalization of the disjunctive graph of Roy and Sussman (1964). In fact, in the disjunctive graph, the pairs of alternative arcs (called disjunctive arcs) are all in the form (i,j), (j,i), where i and j are two operations to be processed on the same machine.

The remaining part of this section briefly illustrates the alternative graph model of some typical constraints arising in scheduling problems. It then describes a complex application of the alternative graph arising in the steel-making industry.

First consider a pair of consecutive operations in a job, say o_i and o_j, and assume that o_j must start processing within k time units after the completion of o_i (i.e. $t_j \leq t_i + p_i + k$). We call it a *perishability* constraint, since it represents the fact that a job deteriorates when stored for more than k time units between the two consecutive operations. We can represent this constraint with a pair of fixed arcs (i,j) and (j,i) having length p_i and $-p_i - k$, respectively. If $k = 0$, we have a tight no-wait constraint (figure 1).

Now consider the blocking constraint used to model the absence of storage capacity between machines. The rest of this paper will distinguish two types of operations: ideal and blocking. We say that an ideal operation o_j remains on machine $M(j)$ from its starting time to completion time and then leaves $M(j)$, which becomes immediately available for processing other operations. On the contrary, a blocking operation o_i may remain on machine $M(i)$ even after its completion time, thus blocking it. More precisely, in our model:

- each blocking operation o_i is associated to the subsequent operation in its job, hereafter called $o_{\sigma(i)}$; and

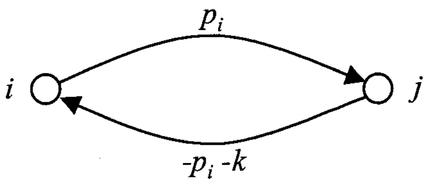

Figure 1. Perishability constraint.

- o_i, having completed processing on machine $M(i)$, remains on it until the machine $M(\sigma(i))$ becomes available for processing $o_{\sigma(i)}$. Machine $M(i)$ therefore remains blocked and it cannot process any other operation until o_i leaves $M(i)$.

The alternative graph model of this constraint is as follows. Consider two operations o_i and o_j, where o_i is blocking, and such that $M(i) = M(j)$. Since o_i and o_j cannot be executed at the same time, we associate with them a pair of alternative arcs. Each arc represents the fact that one operation must be processed before the other. If o_i is processed before o_j, and since o_i is blocking, $M(i)$ can start processing o_j only after the starting time of $o_{\sigma(i)}$, when o_i leaves $M(i)$. We represent this situation with the alternative arc $(\sigma(i), j)$ having length $a_{\sigma(i)j} = 0$. The other alternative arc depends on the fact that o_j can be either blocking on not. If o_j is blocking, then we add the alternative arc $(\sigma(j), i)$ of length $a_{\sigma(j)i} = 0$. If o_j is ideal, then we add the arc (j, i), whose length is $a_{ji} = p_j$. In figure 2(a) and (b), the case of o_j blocking and o_j ideal, respectively, is shown. Here the fixed (alternative) arcs are depicted with solid (dashed) lines.

Figure 3 shows the alternative graph formulation for a blocking flow shop with three jobs and four machines. Note that a job having completed processing on the last machine leaves the system. Hence, the last machine is not blocking.

Besides the cited examples, the alternative graph allows one to formulate situations far more generally that the ones mentioned. Among the others, we cite the specification of non-relaxable earliest–latest starting times for the operations, assembly and disassembly sequences, set-ups, due dates, release times, maintenance operations, material-handling delays and many other operational side constraints.

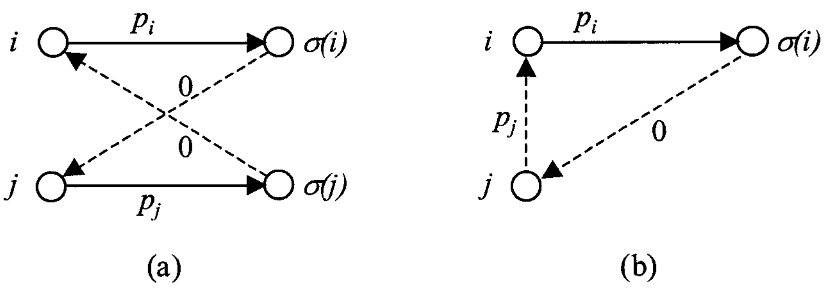

Figure 2. Alternative arcs for a pair of operations o_i and o_j.

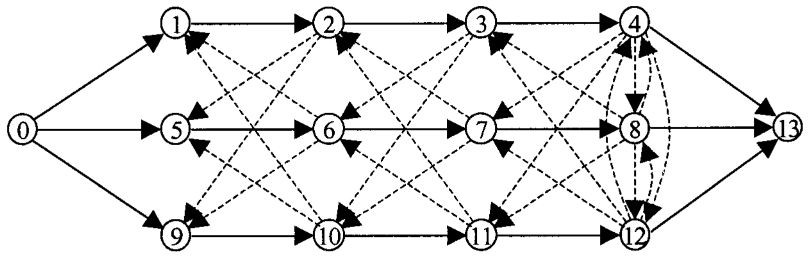

Figure 3. Alternative graph for a blocking flow shop with three jobs and four machines.

See White and Rogers (1990) and Mascis and Pacciarelli (2000) for more general examples.

The rest of this section deals with one such industrial application arising in the production of steel. The purpose of this example is to show how complex scheduling problems can be formulated by means of the alternative graph. Hence, in what follows we will give only a brief introduction and formulate a simplified version of the problem.

3.1. *Production scheduling in a steel-making plant*

High-grade steel-making plants produce a great variety of steel grades to be used in a number of applications. Lot sizes are typically small, and the production process is subject to a number of complex metallurgical requirements in order to achieve suitable properties on the final product. The production-scheduling problem consists of determining the sequencing of operations to be performed on molten steel at the different production stages from steel-making to continuous casting. We will refer in particular to a line for the production of stainless steel in a steel-making plant in Central Italy. Figure 4 shows the layout of the production line. The production process consists of a sequence of high-temperature operations starting with the loading of scrap iron in a electric arc furnace (EAF). The time to melt the scrap iron ranges from 70 to 80 min, including a few minutes for set-up and furnace maintenance operations.

The liquid steel is then poured into ladles, which a crane transports to a subsequent machine, called the argon oxygen decarburation unit (AOD), where nickel, chromium and other elements are added to the steel to meet the chemical quality requirements. Between the EAF and the AOD, there is room for storing up to three ladles, but since the stored steel cools down, it must be reheated in the AOD, adding extra costs. After the AOD, the ladles are transported to a ladle furnace (LF), which can host at most two ladles. Even if some operations are executed in the LF, in practice it acts as a buffer to maintain the ladles at the proper temperature before the last operation, which is execution in the continuous caster (CC). Here the liquid steel is casted and cooled to form slabs. The CC needs to be tooled with a particular tool, the flying tundish, which determines the format of slabs. If two subsequent ladles belong to the same lot, they must be cast without interruption, otherwise the tundish deteriorates and a set-up is necessary to substitute it. However, the tundish must be changed when switching between lots as well as after a given number of identical ladles, which depends on the steel quality. The size of a lot may vary, therefore, from one to seven ladles. The set-up time needs about 60 min, during which time the CC is blocked. Finally, the time horizon for a planning period is 1 week, which corresponds to about an average of 120 ladles (jobs) and 30 lots.

The scheduling problem can be modelled as a permutation flow shop with limited buffers and three additional constraints.

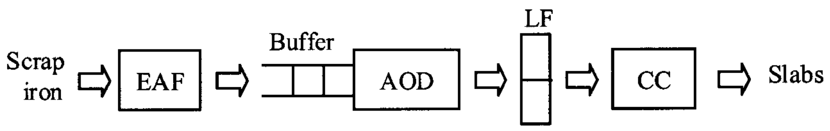

Figure 4. Layout of the stainless steel production line.

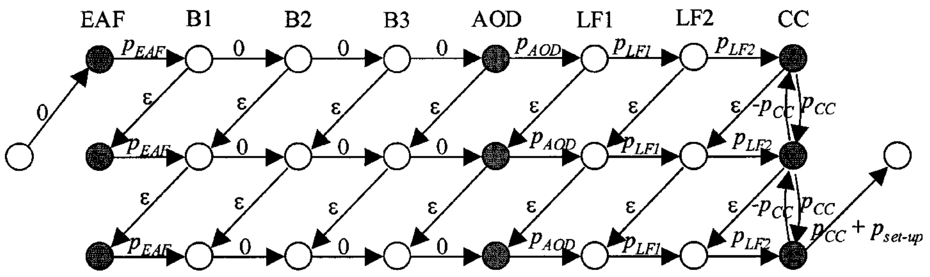

Figure 5. Graph for a lot with three ladles.

- All parts in a lot must be sequenced consecutively.
- There is a sequence-independent set-up time on the last machine (CC) between two consecutive lots.
- There is a no-wait constraint between any two consecutive casting operations in the same lot.

The problem is to find a sequence of lots maximizing the production rate of the plant during 1 week. There is also the secondary goal of reducing the amount of time spent by the ladles in the first buffer, between the EAF and the AOD, even if the primary objective pursued by the schedulers in the plant is to minimize the completion time of all lots.

Figure 5 shows the alternative graph corresponding to a lot with three ladles. The processing of a ladle is a chain of eight nodes. The first one corresponds to the starting time of the heating operations in the EAF, the next three nodes represent the three positions of the subsequent buffer, the fifth node is associated with the processing in AOD unit, the sixth and seventh nodes are associated with the LF, and the eighth is the casting operation. The pair of arcs between two consecutive casting operations represents the no-wait constraint on the CC. The set-up time on the continuous caster is shown in figure 5 by increasing the weight on the arc outgoing from the last node. The diagonal arcs having length ϵ represent the blocking constraints. The weight ϵ represents the fact that all movements in the plant are performed by a single crane. Hence, a ladle can move to the next machine only ϵ time after that the machine has been emptied.

Given a pair of lots, we formulate the capacity constraint of the machines by adding an alternative pair of arcs for each machine (including buffer positions). Note that all machines but the CC here are blocking machines. The complete alternative graph for two lots is shown in figure 6. Here, the first lot has three ladles and the second lot has two ladles. Fixed (alternative) arcs are depicted with solid (dashed) lines. Each pair of alternative arcs represents the constraint that the last operation in a lot to be executed on a given machine must precede the first operation of the other lot to be executed on the same machine, or vice versa.

4. Heuristics

This section deals with a new heuristic procedure for finding good, complete and consistent selections for any given alternative graph.

The first part of this section describes our procedure and compare its results with other greedy algorithms from the literature. The scond part of this section reports on

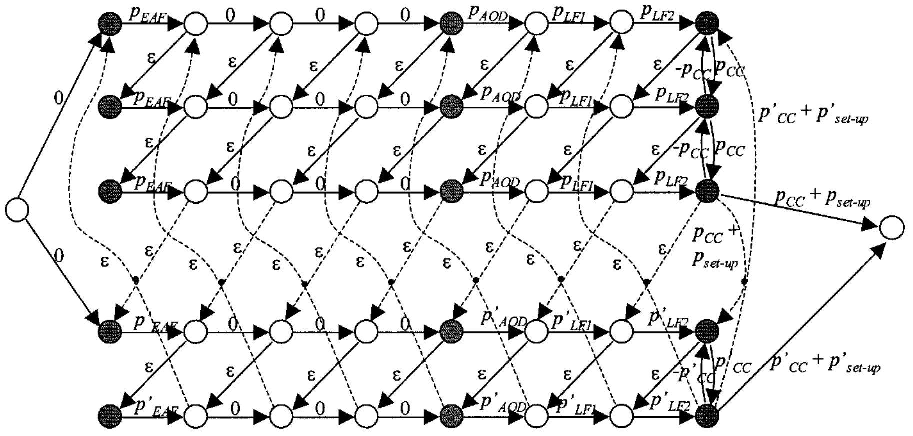

Figure 6. Alternative graph for two lots.

several experiments on real data, referring to the steel-making plant described in Section 3.1.

To describe the procedure, we need the following notation. Given an alternative graph $G = (N, F, A)$, a *selection* S is a set of arcs obtained from A by choosing at most one arc from each pair. The selection is *complete* if exactly one arc from each pair is chosen. Given a pair of alternative arcs $((i,j),(h,k)) \in A$, we say that (i,j) is *selected* in S if $(i,j) \in S$, whereas we say that (i,j) is *forbidden* in S if $(h,k) \in S$. Finally, the pair is *unselected* if neither (i,j) nor (h,k) is selected in S.

Given a selection S, let $G(S)$ indicate the graph $(N, D \cup S)$. A selection S is *consistent* if the graph $G(S)$ has no positive length cycles. Given a consistent selection S, we say that (i,j) is *implied* by S if its alternative arc (h,k) would cause a cycle if added to $G(S)$. An *extension* of S is a complete consistent selection S' such that $S \subseteq S'$, if it exists. With this notation, each schedule is associated with a complete consistent selection on the corresponding alternative graph.

We are now in the position of describing algorithm AMCC (avoid maximum current $C_{\max}$). It is based on the idea of repeatedly enlarging a consistent selection, given a general alternative graph in the form of Problem 3.1, and concluding either with a feasible solution or with a non-consistent selection. In particular, AMCC selects one arc at a time and then adds to the selection all the implied arcs to preserve consistency as long as possible. If both arcs in an unselected pair cause a positive length cycle in $G(S)$, the procedure fails in finding a feasible solution. The step is repeated until a complete selection is built or an inconsistency is detected.

Figure 7 shows the general sketch of the heuristic. We define $l(i,j)$ as the length of the longest path in $G(S)$ from node i to node j. We use the convention if there is no path from i to j, then $l(i,j) = +\infty$.

At each step, AMCC selects an arc (h,k) that, if selected, would improve most the length of the longest path in $G(S)$. Hence, *AMCC* forbids it by selecting its alternative (i,j).

Note that when choosing the pair $((h,k),(i,j)) \in A$, a 'tight' may occur among different arc pairs. Hence, we developed two versions of AMCC. AMCC1 chooses among the pairs fulfilling the condition $l(0,h) + a_{hk} + l(k,n) = \max_{(u,v)\in A}\{l(0,u) + a_{uv} + l(u,n)\}$, the one such that $l(0,i) + a_{ij} + l(j,n)$ is minimum.

```
Algorithm AMCC

Input: Alternative graph $G = (N, F, A)$.

begin
$S = \emptyset$.
While $A \neq \emptyset$ do
    begin
    Choose a pair $((h,k),(i,j)) \in A$, such that:
            $l(0,h) + a_{hk} + l(k,n) = \max_{(u,v)\in A}\{l(0,u) + a_{uv} + l(u,n)\}$.
    Select $(i,j)$, i.e. $S := S \cup \{(i,j)\}$, $A := A - \{((h,k),(i,j))\}$,
    If $\exists((h,k),(i,j)) \in A : l(k,h) + a_{hk} > 0, l(j,i) + a_{ij} > 0$
        then STOP, the procedure failed in finding a feasible solution,
        else while $\{((u,v),(p,q)) \in A : l(v,u) + a_{uv} > 0\} \neq \emptyset$ do
            begin
                $S := S \cup \{(p,q)\}$, $A := A - \{((u,v),(p,q))\}$
                (here all arcs implied by $S$ are selected)
            end
    end
end.
```

Figure 7. Sketch of the algorithm AMCC.

AMCC2 chooses the one such that $l(0,i) + a_{ij} + l(j,n)$ is maximum. In what follows, we call AMCC the best solution found by AMCC1 and AMCC2.

We tested the performance of the heuristics on a sample of academic job shop instances from the literature. Problems Abz5-9 were generated by Adams *et al.* (1988), problems Orb1-10 were generated by Applegate and Cook (1991), problems LA1-40 were generated by Lawrence (1984), and problems Ft6, Ft10 and Ft20 were the three examples proposed in Muth and Thompson (1963).

In table 1 we compare the performance of AMCC over the 58 instances with all greedy procedures for which we could find results in the literature. *Bidir* was proposed in Dell'Amico and Trubian (1993), INSA was proposed in Nowicki and Smutnicki (1996), BJS is the initial heuristic used by Brucker *et al.* (1994) for their branch and bound code, and Dspt is the best dispatching solution over five different list schedules, as reported in Ovacik and Uzsoy (1997). The first column reports the instance name, the second reports the number of machines and jobs of the instance, and the third reports the best known lower bound on the instance. An asterisk indicates that an optimal solution has been found. From columns 4 to 8, the solutions found by the five heuristics are reported, and from columns 9 to 13 the relative errors of the heuristics are computed. The last three rows report the average and maximum errors of the different heuristics over all instances, and the average errors over the 45 common instances, respectively. This computational experience clearly show that AMCC outperforms the other four heuristics.

From the above results, we therefore expect good performance from AMCC whenever it terminates with a feasible solution. Unfortunately, this is not always the case. Hence, in what follows, we are interested in finding some conditions under which AMCC is guaranteed to find a feasible solution. One important case is the well-known academic job shop problem with infinite capacity buffers owing to the well-known property shown in Proposition 4.1.

Instance	Size	LB	AMCC	Bidir	INSA	BJS	Dspt	% AMCC from LB	% Bidir from LB	% INSA from LB	% BJS from LB	% Dspt from LB
Abz5	(10,10)	1234*	1318	1359	1381	1379	1366	6.81	10.13	11.91	11.75	10.70
Abz6	(10,10)	943*	985	1025	1012	1052	1009	4.45	8.70	7.32	11.56	7.00
Abz7	(15,20)	655	753	785	–	–	785	14.96	19.85	–	–	19.85
Abz8	(15,20)	638	783	804	–	–	834	22.73	26.02	–	–	30.72
Abz9	(15,20)	656	777	821	–	–	807	18.45	25.15	–	–	23.02
Ft06	(6,6)	55*	55*	56	59	55*	60	0	1.82	7.27	0	9.09
Ft10	(10,10)	930*	985	1076	994	1090	1088	5.91	15.70	6.88	17.20	16.99
Ft20	(20,5)	1165*	1338	1310	1269	1454	1272	14.85	12.45	8.93	24.81	9.18
Orb01	(10,10)	1059*	1213	1281	–	–	1269	14.54	20.96	–	–	19.83
Orb02	(10,10)	888*	924	1035	–	–	1037	4.05	16.55	–	–	16.78
Orb03	(10,10)	1005*	1113	1214	–	–	1198	10.75	20.80	–	–	19.20
Orb04	(10,10)	1005*	1108	1161	–	–	1223	10.25	15.52	–	–	21.69
Orb05	(10,10)	887*	924	1049	–	–	1041	4.17	18.26	–	–	17.36
Orb06	(10,10)	1010*	1107	–	–	–	1307	9.60	–	–	–	29.41
Orb07	(10,10)	397*	440	–	–	–	473	10.83	–	–	–	19.14
Orb08	(10,10)	899*	950	–	–	–	1132	5.67	–	–	–	25.92
Orb09	(10,10)	934*	1015	–	–	–	1136	8.67	–	–	–	21.63
Orb10	(10,10)	944*	1030	–	–	–	1134	9.11	–	–	–	20.13
La01	(10,5)	666*	666*	724	666*	671	735	0	8.71	0	0.75	10.36
La02	(10,5)	655*	694	739	722	835	756	5.95	12.82	10.23	27.48	15.42
La03	(10,5)	597*	735	710	681	696	671	23.12	18.93	14.07	16.58	12.40
La04	(10,5)	590*	679	664	659	696	677	15.08	12.54	11.69	17.97	14.75
La05	(10,5)	593*	593*	593*	593*	593*	593*	0	0	0	0	0
La06	(15,5)	926*	926*	940	950	926*	926*	0	1.51	2.59	0	0
La07	(15,5)	890*	984	946	976	960	960	10.56	6.29	9.66	7.87	7.87
La08	(15,5)	863*	873	984	868	893	939	1.16	14.02	0.58	3.48	8.81
La09	(15,5)	951*	986	1017	951*	951*	1000	3.68	6.94	0	0	5.15
La10	(15,5)	958*	1009	958*	958*	958*	966	5.32	0	0	0	0.84
La11	(20,5)	1222*	1239	1259	1293	1222*	1222*	1.39	3.03	5.81	0	0
La12	(20,5)	1039*	1039*	1044	1044	1050	1039*	0	0.48	0.48	1.06	0
La13	(20,5)	1150*	1161	1160	1154	1189	1150*	0.96	0.87	0.35	3.39	0
La14	(20,5)	1292*	1305	1294	1328	1292*	1292*	1.01	0.15	2.79	0	0
La15	(20,5)	1207*	1369	1304	1323	1363	1343	13.42	8.04	9.61	12.92	11.27
La16	(10,10)	945*	979	1045	1077	1074	1066	3.60	10.58	13.97	13.65	12.80
La17	(10,10)	784*	800	838	821	849	842	2.04	6.89	4.72	8.29	7.40
La18	(10,10)	848*	916	973	926	926	962	8.02	14.74	9.20	9.20	13.44
La19	(10,10)	842*	846	941	971	977	947	0.48	11.76	15.32	16.03	12.47
La20	(10,10)	902*	930	1050	1003	987	980	3.10	16.41	11.20	9.42	8.65
La21	(15,10)	1046*	1241	1210	1179	1175	1270	18.64	15.68	12.72	12.33	21.41
La22	(15,10)	927*	1032	1087	1032	1060	1141	11.33	17.26	11.33	14.35	23.09
La23	(15,10)	1032*	1131	1093	1132	1155	1157	9.59	5.91	9.69	11.92	12.11
La24	(15,10)	935*	999	1132	1021	1085	1110	6.84	21.07	9.20	16.04	18.72
La25	(15,10)	977*	1071	1175	1147	1086	1172	9.62	20.27	17.40	11.16	19.96
La26	(20,10)	1218*	1378	1355	1397	1342	1425	13.14	11.25	14.70	10.18	17.00
La27	(20,10)	1235*	1353	1470	1466	1413	1488	9.55	19.03	18.70	14.41	20.49
La28	(20,10)	1216*	1322	1415	1485	1468	1551	8.72	16.37	22.12	20.72	27.55
La29	(20,10)	1130	1392	1401	1385	1352	1344	23.19	23.98	22.57	19.65	18.94
La30	(20,10)	1355*	1476	1493	1463	1577	1547	8.93	10.18	7.97	16.38	14.17
La31	(30,10)	1784*	1871	1840	1966	1903	1935	4.88	3.14	10.20	6.67	8.46
La32	(30,10)	1850*	1942	1928	1982	1850*	1881	4.97	4.22	7.14	0	1.68
La33	(30,10)	1719*	1897	1802	1767	1766	1870	10.35	4.83	2.79	2.73	8.78
La34	(30,10)	1721*	1934	1837	1844	1825	1890	12.38	6.74	7.15	6.04	9.82
La35	(30,10)	1888*	2017	1983	1967	2028	2083	6.83	5.03	4.18	7.42	10.33
La36	(15,15)	1268*	1347	1383	1445	1406	1547	6.23	9.07	13.96	10.88	22.00
La37	(15,15)	1397*	1547	1657	1726	1588	1649	10.74	18.61	23.55	13.67	18.04
La38	(15,15)	1196*	1342	1362	1307	1371	1503	12.21	13.88	9.28	14.63	25.67
La39	(15,15)	1233*	1361	1502	1393	1442	1420	10.38	21.82	12.98	16.95	15.17
La40	(15,15)	1222*	1340	1389	1387	1417	1422	9.66	13.67	13.50	15.96	16.37
AVRG								8.33	11.86	9.24	10.12	13.95
W.C.								23.19	26.02	23.55	27.48	30.72
AVRG45								7.54	10.34	9.24	10.12	11.65

Table 1. Comparison among AMCC and four heuristics from the literature.

Proposition 4.1: *Given any instance of the job shop problem with infinite capacity buffers, for any consistent selection always exists an extension.*

Proof: To prove Proposition 4.1, it is sufficient to observe that given a formulation of the job shop problem with infinite capacity buffers, a consistent selection S and an unselected pair of alternative arcs $((i,j),(j,i)) \in A$, it is possible to obtain a larger consistent selection by adding to S one of the two alternative arcs.

Consider the two selections: $S' = S \cup \{(i,j)\}$ and $S'' = S \cup \{(j,i)\}$. If S' is not consistent, then the graph $G(S')$ contains a cycle including arc (i,j), and therefore $G(S)$ contains a directed path from j to i. If S'' is not consistent, then the graph $G(S'')$ contains a cycle including arc (j,i), and therefore $G(S)$ contains a directed path from i to j. These two paths form a cycle in $G(S)$, which contradicts the hypothesis that S is consistent. By repeatedly applying this argument it is possible to obtain an extension of S. □

This key property does not hold for a more general situation, e.g. for blocking problems. A simple example of this fact is given by the flow shop with two jobs and three blocking machines (figure 8). Here, operations o_1 and o_4 have to be processed on the same machine, i.e. $M(1) = M(4)$. Moreover, $M(2) = M(5)$, $M(3) = M(6)$, while the start and finish operations, o_0 and o_7, respectively, are dummy operations. The selection $S = \{(2,4),(6,3)\}$ is consistent but no extension exists in this case since it is not possible to select one arc from the pair $((3,5),(6,2))$ without creating a positive length cycle in the graph. As a consequence of this fact, we observe that an algorithm based on the idea of repeatedly enlarging a consistent selection may fail to find a feasible solution.

On the other hand, from a closer look to the algorithm of figure 7, we can guarantee that AMCC terminates with a feasible solution under a weaker property than the one in Proposition 4.1. The property is the following: for any consistent selection S in which all arcs implied by S are selected there always exists an extension. Note that the selection in figure 8 does not satisfy this property since the unselected arc $(3,5)$ is implied by $(2,4)$ and $(6,2)$ is implied by $(6,3)$. In fact, in this problem, by selecting only one arc, say $(2,4)$, and the implied arcs $(3,5)$ and $(3,6)$, we immediately get a feasible solution. This is always the case in the blocking flow shop problems. Also, in the steel-making problem of Section 3.1, we can ensure that AMCC always finds a feasible selection and therefore we can expect a good performance in this case.

We tested AMCC on real data provided by a steel company in Central Italy. The weekly production plan requires the sequencing of approximately 130 ladles,

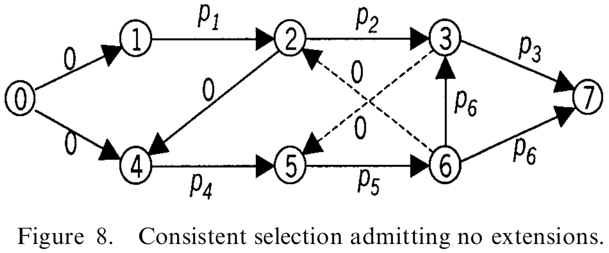

Figure 8. Consistent selection admitting no extensions.

Implemented sequence (simulated)	AMCC (simulated)	AMCC (evalulated on the graph)	Lower bound (evaluated on the graph)
9328	9138	9126	9126

Table 2. Computational experiments on the real data.

grouped into 35 lots ranging from one to seven ladles each. We compared the resulting production plan with the one actually implemented in the plant, and produced by hand. The comparison was carried out on the basis of a detailed simulator used in the plant to plan the activities. Since the real processing times of the operations are partially stochastic, we used in the experiments the means for each operation. These are the values used in the plant to produce the usual weekly plan.

The results of this comparison are shown in table 2. All values are expressed in minutes. The first column shows the makespan of the sequence produced by hand, while the second column shows the makespan obtained by AMCC. Both data are evaluated with the plant simulator. The third column shows the makespan obtained by AMCC, which was computed by the algorithm AMCC. We observe that the makespan evaluated on the alternative graph is extremely close to the makespan evaluated by the simulator. We conclude that the alternative graph model of the scheduling problem includes all the relevant details of the industrial problem. The fourth column shows a lower bound on the makespan, which was computed by using the lower bound of Mascis and Pacciarelli (2000), which proves that AMCC found the optimal solution in this case.

From the comparison of the lower bound with the makespan of the sequence implemented in the plant, it follows that the quality of this sequence is within 2% of the optimum. Hence, there is little space to improve it and it can be concluded that the use of approximated models is unlikely to produce good results in the steelmaking process. On the other hand, with the detailed model given by the alternative graph, AMCC can produce a better sequence than the real one. Moreover, while the computing time of AMCC for 35 lots and 130 ladles is limited to a few minutes, the real sequence is produced by many human-hours. The production sequence produced by AMCC was also analysed by the industrial partner that confirmed its validity.

5. Conclusions

A wide variety of practical scheduling problems have been shown that can be modelled by means of the alternative graph. A new fast algorithm was also given that is able to find feasible solutions for many practical problems. Unfortunately, for general alternative graphs, AMCC may fail to find a feasible solution, while it is very promising when solving those scheduling problems where it is possible to guarantee that it terminates with a feasible solution. Relevant academic cases in which this happens are the job shop problem with infinite capacity buffers, and the blocking flow shop problem. More importantly, there are many practical scheduling problems in which AMCC always terminates with a feasible solution. The paper showed a case study arising in the steel industry, but in general this in true for most industrial flow lines with limited capacity buffers, even in the presence of additional side constraints. However, further research is needed to develop effective solution algorithms for more general problems such as blocking or no-wait job shop problems, and to develop better heuristics and exact solution algorithms.

References

Adams, J., Balas, E. and Zawack, D., 1988, The shifting bottleneck procedure for job shop scheduling. *Management Science*, **34**, 391–401.

Applegate, D. and Cook, W., 1991, A computational study of the job shop scheduling problem. *Orsa Journal on Computing*, **3**, 149–156.

Arbib, C., Pacciarelli, D. and Smriglio, S., 1999, A three-dimensional matching model for perishable production scheduling. *Discrete Applied Mathematics*, **92**, 1–15.

Balas, E., 1979, Disjunctive programming. *Annals of Discrete Mathematics*, **5**, 3–51.

Brucker, P., Jurish, B. and Sievers, B., 1994, A branch and bound algorithm for the job-shop scheduling problem. *Discrete Applied Mathematics*, **49**, 107–127.

Carlier, J. and Pinson, E., 1994, Adjustment of heads and tails for the job-shop problem. *European Journal of Operational Research*, **78**, 146–161.

Dell'Amico, M. and Trubian, M., 1993, Applying taboo search to the job-shop scheduling problem. *Annals of Operations Research*, **41**, 231–252.

Grabowski, J., Pempera, J. and Smutnicki, C., 1997, Scheduling in production of concrete wares. In *Operations Research Proceedings 1996 (Braunschweig)* (Berlin: Springer), pp. 192–196.

Hall, N. J. and Sriskandarajah, C., 1996, A survey on machine scheduling problems with blocking and no-wait in process. *Operations Research*, **44**, 510–525.

Lawrence, S., 1984, *Supplement to 'Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques'* (Pittsburgh: GSIA, Carnagie Mellon University).

Lixin Tang, Jiyin Liu, Aiying Rong and Zihou Yang, 2000, A mathematical programming model for scheduling steelmaking-continuous casting production. *European Journal of Operational Research*, **120**, 423–435.

Mascis, A. and Pacciarelli, D., 2000, Machine scheduling via alternative graphs. Report DIA-46-2000 (Rome: Dipartimento di Informatica e Automazione, Università Roma Tre).

McCormick, S. T., Pinedo, M. L., Shenker, S. and Wolf, B., 1989, Sequencing in an assembly line with blocking to minimize cycle time. *Operations Research*, **37**, 925–935.

Muth, J. F. and Thompson, G. L., (eds), 1963, *Industrial Scheduling* (Dordrecht, The Netherlands: Kluwer).

Nahmias, S., 1982, Perishable inventory theory: a review. *Operations Research*, **30**, 680–708.

Nowicki, E. and Smutnicki, C., 1996. A fast taboo search algorithm for the job shop scheduling problem. *Management Science*, **42**, 797–813.

Ovacik, I. M. and Uzsoy, R., 1992, A shifting bottleneck algorithm for scheduling semi-conductor testing operations. *Journal of Electronic Manufacturing*, **2**, 119–134.

Ovacik, I. M. and Uzsoy, R., 1997. *Decomposition Methods for Complex Factory Scheduling Problems* (Englewood Cliffs, NJ: Prentice-Hall).

Pinedo, M., 1995, *Scheduling—Theory, Algorithms and Systems* (Englewood Cliffs, NJ: Prentice-Hall).

Reklaitis, G. V., 1982, Review of scheduling of process operations. *AIChE Symposium Series*, **78**, 119–133.

Roy, B. and Sussman, B., 1964, Les problèm d'ordonnancement avec contraintes disjonctives. Note DS No. 9bis (Paris: SEMA).

Sadeh, N., Sycara, K. and Xiong, Y., 1995, Backtracking techniques for the job shop scheduling constraints satisfaction problem. *Artificial Intelligence*, **76**, 455–480.

Schutten, J. M. J., 1998, Practical job shop scheduling. *Annals of Operations Research*, **83**, 161–177.

Strusevich, V. A., 1997, Shop scheduling problems under precedence constraints. *Annals of Operations Research*, **69**, 351–377.

White, K. P. and Rogers, R. V., 1990, Job-shop scheduling: limits of the binary disjunctive formulation. *International Journal of Production Research*, **28**, 2187–2200.