Scheduling problems for logistic platforms with fixed staircase component arrivals and various deliveries hypotheses

Susana Carrera, Wahiba Ramdane-Cherif, Marie-Claude Portmann

Scheduling problems for logistic platforms with fixed staircase component arrivals and various deliveries hypotheses

Susana Carrera[1], Wahiba Ramdane-Cherif[1], Marie-Claude Portmann[1],

[1] INPL-LORIA, ENSMN, Parc de Saurupt, CS 14234, 54042 Nancy, France
{carrersu, ramdanec, portmann}@loria.fr

Abstract. This paper studies the scheduling problem within a logistic platform node. It corresponds to the last node in the supply chain located just before the final customers. In this node, customer orders are prepared using known quantities of components received from the suppliers. For upstream flows, trucks deliver known component quantities at fixed times. For downstream flows, optimized customer delivery tours are planned at fixed times except the last one which will be done at a flexible time corresponding to the end of the schedule. We consider a simplified case of one bottleneck-assembling machine. Several sub cases are considered depending on the number of the fixed and flexible deliveries and also on the chosen criteria. For each sub case, upper and lower bounds are proposed and compared experimentally on heterogeneous instance families. All the ingredients are therefore available to design a branch and bound for these new scheduling problems.

Keywords: Scheduling, Supply Chain, Consumable Resources, Several Deliveries, Tardiness Criteria.

1 Introduction

This study has been inspired by a real platform logistic problem arising at a "shoes firm". The firm owns the platform, a vehicle fleet and about a hundred stores. This firm completely manages the platform work and the downstream supply chain but it does not take in charge the upstream flows. In previous work we tackled with predictive planning problem corresponding to two seasonal volume peaks (spring and autumn). We proposed generic integer linear programming models in order to smooth the platform workload using negotiations of dates and delivered quantities.

We considered the upstream flows as being slightly flexible at the planning level, but at the operational level, studied here, the upstream flows are fixed and the arrival of products can be represented by known cumulated staircase curves. On the other hand, the firm owns the stores and hence it can manage completely the vehicle routing organization between the platform and the stores. In order to minimize the vehicle routing cost, optimized tours can be computed for subset of stores (grouped by regions) and ideal dates and quantities can be associated to planned tours.

Each job of our scheduling problem corresponds to the preparation of a store order. One order contains a list of items. A list composed of quantity, category, type, size and eventually a colour is associated to every item. The aim is to schedule the preparation of stores' orders under upstream (component arrival) and downstream (ideal delivery dates) constraints.

Another application can be an illustration of our scheduling problems. A small company located near Nancy town receives various magazines and advertising documents; those articles are grouped and wrapped into plastic, then the packages are completed with gadgets and customers addresses. Every day, the packages are sent by the post office at a fixed time (4pm). A second private truck will leave lately at a flexible time depending on the end of the daily urgent work. Some jobs have very precise delivery date and must be included in one fixed delivery while other ones have flexible delays.

Relevant scheduling literature is analyzed in section 2. In section 3, several specific scheduling problems are defined. In sections 4 and 5 rapid approximation methods and lower bounds are respectively proposed. Computational evaluations are given in section 6. Finally, section 7 contains a conclusion and some suggestions for further researches.

2 State of the art

Our problem is related to resource constrained scheduling literature and scheduling under delivery date constraints literature.

In most of the scheduling problems involving resource constraints, two types of resources are distinguished: renewable and non-renewable (or consumable) resources. Since the components in the supply chain node are considered as consumable, we will focus on this type of resources in our literature analysis. Blazewicz J. *et al.* 1986 identified two problems where the allocation of constrained resources has been mainly considered: resource constrained project scheduling problem (RCPSP) and machine scheduling.

Carlier J. 1984 associated consumable resources to financial resources because of the similarity of the availability constraints they must satisfy. He proves that RCPSP with financial constraints and arbitrary precedence constraints is polynomial if no renewable resource is considered; it is NP-hard if there are consumption and production of consumable resources. Patterson *et al.* 1989 and Carlier J. *et al.* 2009 dealt with the project scheduling problem where the units of resources are produced or consumed at the occurrence of precedence-related events. They proposed a list-scheduling based algorithm to minimize the makespan.

Carlier J. 1984 also provided a variety of complexity results on non preemptive one machine scheduling subject to financial constraints. The financial constrained one machine scheduling problem is NP-hard when the job processing times are not equal to one. Slowinski R. 1984 handled the preemptive job scheduling on parallel machines with financial constraints. The consumption rate of financial resource is

constant during job processing. Slowinski proposed a two-phase method based on linear programming to minimize the schedule length.
Cochand M. *et al.* 1989 took into account consumable resources with a time-varying supply (i.e. staircase and piecewise linear). They generalized the two-phase method to consider this type of resources. Toker A. *et al.* 1991 and Xie J. 1997 studied the case of one machine scheduling with one or several financial resources which are continuously supplied at a constant rate. This problem is polynomial and it is solved by Johnson rule for the two-machine flow shop scheduling problem. Recently, Gafarov E.R. *et al.* presented some complexity results for several objective functions: makespan, total tardiness, number of tardy jobs, total completion time and maximum lateness.

Concerning scheduling with several delivery dates, Matsuo H. 1988 introduced an environment in which delivery dates are fixed and given a-priori before any jobs are processed. He proved that the total weighted tardiness and the total tardiness problems with fixed shipping dates are NP-hard. In Hall N.G. *et al.* 2001, the authors provided either a polynomial time algorithm or a proof of intractability of problems with fixed delivery dates for a variety of objectives.

This paper proposes to combine two scheduling areas: consumable resources for the upstream flows (arrivals) and several delivery dates. To the best of our knowledge, this problem is new and has never been considered before.

3 Problem definition

A set N of n independent jobs corresponding to order preparation must be processed by the platform. Preemption of the jobs is assumed not to be allowed. Any renewable resource (human or material) considered can handle only one job at a time. Each job is assumed to be ready for processing when all the necessary components to execute it are available. These components are then consumed. We denote by p_j (> 0) the processing time of job j. Each job $j \in N$ consumes $a_{j,k}$ (≥ 0) units of component k at the starting time of job j. The components can arrive at different dates with fixed supplier deliveries. The arrival of each component can be represented as a fixed cumulated staircase curve. We denote by *Stairs*(nc), nc consumable resources whose cumulated arrival curves are staircases, see Fig. 1.
In this paper, we make various assumptions for the deliveries. In the most general case, we have f fixed delivery dates $D_1, D_2 \ldots D_f$, each of which corresponds to an optimized tour associated to a sub set of customers. Eventually, at a last flexible delivery date greater than D_f, the remaining jobs could be distributed at the end of the schedule (C_{max}). At each delivery date D_d, any already prepared order corresponding to the sub set of associated customers will be loaded into truck(s). We do not consider limit of capacity for each delivery, this will be discussed in the perspectives. As in Hall N.G. et al (2001), we denote by $\hat{C}_j$ the date at which the job j is delivered. It corresponds to the first delivery date which follows its completion time C_j. A penalty w_j is associated to each job j and represents an estimation of the loss of sales per time

unit when orders arrive tardy to the stores.

In this paper, we assume that every tour visits the whole set of customers and we consider three hypotheses for the delivery dates, see Fig. 1. Under hypothesis (A), there exists only one flexible delivery date at the end of the last job ($f = 0$). It corresponds in fact to the well-known C_{max} criterion ($\hat{C}_j = C_{max}$ pour tout j). Hence, every job is considered as late and the penalty cost is proportional to C_{max}.

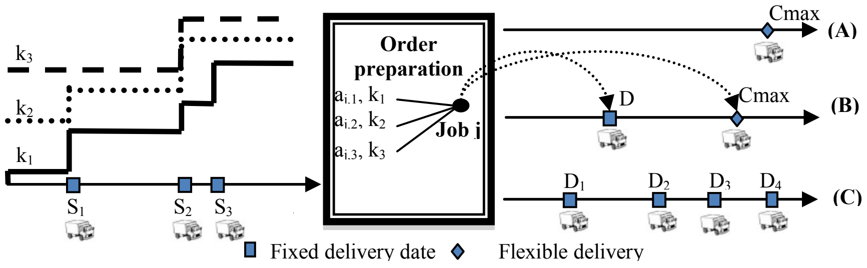

Fig. 1. Delivery hypotheses.

Under hypothesis (B), there are two deliveries, one at a fixed date D_1 ($f = 1$) and one at a flexible date at the end of the schedule (C_{max}). D_1 can be considered as a fixed common due date and any job not sent at date D_1 will be considered as late. For each job j we define $\hat{u}_j = 1$ if job j is late and $\hat{u}_j = 0$ otherwise. The used criterion will be proportional to w_j and to ($C_{max} - D_1$). Since D_1 is a constant, ($C_{max} - D_1$) can be replaced by C_{max} in the criterion. Moreover $\hat{C}_j = D_1$ if $C_j \le D_1$ and $\hat{C}_j = C_{max}$ otherwise.

For assumption (C), $f \ge 2$ (in this paper $f = 4$) and D_f is assumed to be great enough such that $D_f \ge C_{max}$. Here, $\hat{C}_j = D_d$ where D_d is the first delivery date after C_j. Every job is considered as late and must be delivered as soon as possible.

The $\alpha/\beta/\gamma$ notations for the resulting NP-hard scheduling problems are:

(A): $1/Stairs(nc),a_{j,k}/C_{max}$;

(B): $1/Stairs(nc),a_{j,k},D_1/(1+\Sigma w_j\hat{u}_j)C_{max}$;

(C): $1/Stairs(nc),a_{j,k},D_1...D_f/\Sigma w_j\hat{C}_j$.

To the best of our knowledge, except for hypothesis (A), these scheduling problems were not yet considered in the literature.

4 Rapid approximation methods

Our aim is to design rapid approximation methods to be included into branch and bound approaches. We will use list algorithms based on priority orders in order to build active and/or no delay schedules. This section is divided into two parts. In the first one we recall how to build active and no delay schedules in the framework of the considered problem. In the second one we propose a series of specific priority rules.

Active schedule generator: A schedule is said to be active if no operation can be scheduled sooner without delaying at least another operation. In other words, no idle time is created which can contain completely an operation scheduled later. The well-known generic algorithm for active schedule using a given order *σ*: *HAσ* is given in algorithm a).

No delay schedule generator: A schedule is said to be no delay if no machine remains inactive while a job is waiting to be processed on this machine (here with the available components).The well-known generic algorithm for no delay schedule using a given order *σ*: *HNDσ* is given in algorithm b).

To build orders or priority rules, three intuitive and heuristic techniques can be used which are antagonistic:

– in order to keep the resource utilization cumulated curves (*CCCU*) under the resource arrival cumulated curves (*CCCA*) without delaying too much the jobs, it seems better to put first the longest jobs asking for the smallest quantities of components. This can be done by using for example the decreasing order of the processing times (*LPT*) divided by the Sum of the Components Demands (*SCD*). This order will be denoted by *LPT-SCD*. It is to be noted that even for only one component, the order corresponding to decreasing values of processing times divided by the component demand does not minimize C_{max}.

– In order to minimize the sum of tardiness penalties, it is better to put first the shortest jobs with the greatest penalties. This corresponds to the well know *WSPT* order. Without taking into account the *CCCU* and *CCCA* curves, it is a good heuristic for the knapsack problem.

– In order to minimize C_{max}, minimizing the idle times seems a good idea, which leads to prefer no delay schedule, but they are not dominant for regular criteria.

We will use the previously defined orders either individually, or hierarchically (the second order is used only to break ties on the first one), or even dynamically. A dynamical use of orders consists in starting with a given order and continuing with another one when some condition is verified such as $t \geq D_f$ or t greater than the last component arrival. In particular, *WSPT* becomes useless when $t \geq D_f$ and *LPT-SCD* becomes useless after the last component arrival. It is to be noted that if both $t \geq D_f$ and t greater than the last component arrival then any job order can be used to complete the schedule; it is applied for any designed heuristic and for any hypotheses.

We get the following interesting orders:

σ = 1: □ *LPT-WSCD/WSPT,* σ = 3: □ *DYN(LPT-SCD→WSPT)/WSPT*
σ = 2: □ *WSPT/LPT-WSCD,* σ = 4: □ *DYN(WSPT→LPT-WSCD)/LPT-WSCD,*

Algorithm a)

```
Let L be the job list sorted in σ order.
t=0.
for each component i do
    CCCA_i= cumulated curve of component i arrivals
    CCCU_i=[0, 0, …, 0] = cumulated curve of component utilization
endfor
while L is not empty do
    MinC = + ∞
    for each job j do
```

Algorithm b)

```
Let L be the job list sorted in σ order.
t=0.
for each component i do
    CCCA_i= cumulated curve of component i arrivals
    CCCU_i=[0, 0, …, 0] = cumulated curve of component utilization
endfor
while L is not empty do
    MinS = + ∞.
    No-idle-time = false.
```

```
        Place job $j$ as soon as possible after time t so that the modified $CCCU$ curves remained smaller or equal to the $CCCA$ curves.
        Let $S_j$ and $C_j$ be respectively the corresponding starting and completion time of job $j$.
        $MinC = min(MinC,C_j)$.
    endfor
    Let be $j_o$ the first job of list $L$ such that $S_{jo} < MinC$.
    Schedule $j_o$ between $S_{jo}$ and $C_{jo}$.
    Modify the curves $CCCU$.
    Remove $j_o$ from list $L$.
    $t = C_{jo}$.
endwhile
```

```
    for each job $j$ of $L$ until no-idle-time = true do
        Place job $j$ as soon as possible after time $t$ so that the modified $CCCU$ curves remained smaller or equal to the $CCCA$ curves.
        Let $S_j$ and $C_j$ be respectively the corresponding starting and completion time of job $j$.
        $MinS = min (MinS,S_j)$.
        if $MinS = t$ then no-idle-time = true
        endif.
    endfor
    let be $j_o$ the first job of list $L$ such that $S_{jo} == MinS$.
    schedule $j_o$ between $S_{jo}$ and $C_{jo}$.
    modify the curves $CCCU$.
    remove $j_o$ from list $L$.
    $t = C_{jo}$.
endwhile
```

This provides us with at least 4 orders, which can be combined with the two presented generators in order to design 8 rapid approximation methods. While each method is rapid, we will apply the whole set of heuristics and call *HBEST* the global heuristic consisting in applying successively all the methods and keeping the best solution obtained for the considered criterion.

5 Lower Bounds

5.1 Lower bounds using "agreeable orders"

Hypotheses (A) and (B): lower bound for Cmax

Lemma 1: When any job requires exactly the same number a_k of component for each component k, then *LPT* minimizes C_{max}.

The proof is quite obvious by using a series of pairwise exchange on a schedule, which does not verify the order *LPT*. Under slightly different hypotheses: the component arrivals are uniform and continuous; this lemma was proposed by Xie J. 1997.

Lemma 2: If there exists only one component o and the processing times of all jobs are identical (equal to p), then the increasing order *CC* of the component requirement minimizes C_{max}.

The proof is quite identical to the proof of Lemma 1 by using pairwise exchanges on a schedule, which does not verify the order *CC*.

Even if there is only one component *o*, for different processing times and component requirements, the increasing order of the component requirements divided by the processing times does not minimize the C_{max}; in consequence we will use the usual technique called "agreeable orders" to get our lower bound.

If there are several components, we will use the technique of "agreeable orders" independently for each component. This will provide us a lower bound for each component and we will keep the maximal value obtained as the lower bound associated to "agreeable orders".

Considering only one component *o*, let be Π the problem studied in this paper and let Π'(o) a relaxation of the problem Π obtained as follows: the first job of Π'(o) gets the greatest processing time and the smallest component requirement, the second job of Π'(o) gets the greatest remaining processing time and the smallest remaining component requirements and so on until the last job. The jobs of Π'(o) are sorted simultaneously by decreasing values of processing times and increasing values of component requirements and the following Lemma 3 can be applied**.**

Lemma 3: If the decreasing order of processing times *LPT* is identical to the increasing order of component requirement *CC*, then the orders are "agreeable" and minimize the makespan C_{max}.

This lemma is a consequence of lemmas 1 and 2. The solution of problem Π'(o) obtained by using lemma 3 provides us with a lower bound for C_{max} of the initial problem Π .The maximum of the lower bounds obtained for C_{max} by this technique for the whole set of components will be denoted by $LB|AGO|CT$ (lower bound with agreeable orders for last job completion time).

Hypothesis (B): lower bound for the weighted penalty sum of late jobs

Lemma 4: For any interval [0, H], the sum of the included idle times contained in any schedule built by using the agreeable order of lemma 3 is a lower bound of the sum of the included idle times for any feasible schedule on the interval [0,H].

Lemma 4 is a consequence of lemma 3. Let $UP/D_l = D_l - LB|AGO|IT(D_l)$ the difference between D_l (date of departure of the fixed delivery) and $LB|AGO|IT(D_l)$, the greatest lower bound of the idle times on [0, D_l], i.e. an upper bound of the sum of processing times, which can be put on [0, D_l] by taking into account the component arrivals.

To get an upper bound of the weighted penalty sum of on time jobs, we must put on the interval [0,UP/D_l] a subset of jobs, whose sum of processing times is smaller than UP/D_l and sum of penalties w_j is maximized. This problem is equivalent to a knapsack problem with only one constraint, in which the satisfaction values are the tardiness penalties, the weight values are the processing times and the capacity of the knapsack is equal to UP/D_l. To get an upper bound for this knapsack problem, we use the improved upper bound proposed by Martello, S. and Toth, P. 1990. The lower bound of the weighted sum of tardy jobs denoted by $LB|AGOK|ST$ (lower bound for sum of tardiness penalties using agreeable orders and Knapsack relaxations) is obtained by subtracting the obtained upper bound from the total sum of tardiness penalties.

Hypothesis (C): lower bound for the weighted sum of completion times adjusted to delivery times

The process used for hypothesis (B) can be easily extended to this hypothesis. First, we can compute the UP/D_k corresponding to the upper bound of the available time on the interval [0, D_k] by deleting a lower bound of idle times from this interval. Second, we can compute UP/SW_k, an upper bound of the sum of the penalty weight of the jobs which can be put on the interval [0, UP/D_k] by using the upper bound proposed by Martello, S. and Toth, P. 1990. Finally, $UPW_1 = UP/SW_1$ is the penalty sum of jobs assumed to be delivered at date D_1, $UPW_2 = \{UP/SW_2 - UP/SW_1\}$ is the penalty sum of jobs assumed to be delivered at date D_2, …, $UPW_{k+1} = \{UP/SW_{k+1} - UP/SW_k\}$ is the penalty sum of jobs assumed to be delivered at date D_{k+1}, until any jobs are delivered at D_f or sooner. The lower bound of the weighted completion times adjusted to delivery times is equal to the sum of the $UPW_k.D_k$. This lower bound is denoted by $LB|AGOK|\Sigma CT$.

5.2 Lower bounds using integer relaxations

Hypotheses (A) and (B): Relaxation using preemption and continuous consumption of components for *Cmax*

A relaxation can be obtained by accepting job preemption and by assuming that the components are not required at the start of the jobs, but are uniformly and continuously consumed during the job execution. The obtained problem is a particular case of the problem considered by Cochand M. *et al.* 1989, which use linear programming to solve it. The authors used a two-phase procedure, but only the first phase is used here because getting a feasible solution for their problem is useless for computing our lower bound.

The time axis is divided into h periods: $T^1, T^2 \ldots T^h$. The beginning and the end of each period correspond to events. An event is either an arrival of components and/or a departure of a delivery. No event occurs during one period. A linear program is used in which the variable $X_{j,t}$ determines the time quantity of job *j* processed during period *t*. Two families of constraints are designed: the first family limits the sum of processing times associated to one period to the length of the period; the second one implies that the cumulated curves of component consumption are always smaller than the component arrivals. The objective is to minimize the makespan.

It is to be noted that, as in sub section 5.1, the length of each interval [0, T^t] can be adjusted by taking into account a lower bound of the sum of idle times on this interval. This provides us with tighter constraints for the sum of processing times on each interval.

The lower bound for C_{max} obtained by applying the method described below on this relaxed problem will be denoted by $LB|IT\sim IR|CT$ (lower bound using idle times and integer relaxation for last job completion time). This lower bound dominates the lower bound $LB|AGO|CT$ (because the sum of idle times can only increase if we add more constraints) and the lower bound $LB|IR|CT$ in which the integer relaxation is used without taking into account the idle times obtained using agreeable orders without preemption.

Hypotheses (B) and (C): Relaxation using preemption and continuous consumption of components for the weighted sum of penalties

For hypothesis (B), we can use the same relaxation as for C_{max} by adapting the linear program in order to get an upper bound of the sum of on time job penalties. We have only to cut the horizon at time D_1 and to modify the objective function, which consists now in maximizing the penalties associated to the proportion of jobs scheduled on interval [0, UP/D_1]. The constraints on the component arrivals are obviously kept. As previously, this upper bound for on time jobs is transformed in lower bound for late jobs.

Under hypotheses (C), we are in the general case in which events correspond to arrivals and/or to delivery departures. On one hand, the second family of constraints corresponding to the respect of the component arrivals and consumptions are similar to hypotheses (A) and (B). On the other hand, on each interval [0, D_k], the sum of the processing times must be smaller than the usable time UP/D_k. While in the criterion, the sum of the processing times included in the periods between D_{k-1} and D_k are multiplied by D_k.

The lower bounds obtained with this relaxation are denoted by *LB*|*IT~IR*|*ST* and *LB*|*IT~IR*|Σ*CT* (lower bound using idle times and integer relaxation for the sum of tardy job penalties or for the weighted sum of completion times adjusted to deliveries). It is to be noted that there are no domination between *LB*|*IT~IR*|*ST* and *LB*|*AGOK*|*ST* and between *LB*|*IT~IR*|Σ*CT* and *LB*|*AGOK*|Σ*CT* because the relaxations are complementary. In AGOK, the pre-emption is forbidden, but the association between the processing times and the penalties are modified to become agreeable. In IT~IR, the association between times and penalties is not modified while the pre-emption is authorized. In consequence, they can be better or worse depending on the considered instances.

6 Computational Experience

For each generated instance, the horizon length can be estimated by using one of the upper bound of the C_{max} value. We use two parameters to differentiate instance families. Dispersion of the component arrival dates is valuable for any hypotheses. Component arrivals can be dispersed over the time horizon (denoted by DA) or relatively grouped at the beginning of the horizon (denoted by RA). For hypothesis (B), a second parameter is defined: position of the on time delivery date D_1. The on time delivery date is generated either at the middle of the horizon (denoted by MD) or at the third quarter of the horizon (denoted by GD). This gives us four sets of instances denoted by RA/GD, RA/MD, DA/GD and DA/MD. Each set of instances contains currently 20 instances (n = 10 or 20 or 50 jobs). For hypotheses (A) and (C) we group families RA/GD (resp. DA/GD) and RA/MD (resp. DA/MD) in only one family of 40 instances RAD (resp. DAD).

The lower bounds for C_{max} (CT) and for the Weighted Sum of Tardiness (ST): 1 + $\Sigma w_j \hat{u}_j$ are compared in Table 2. The performance is given in each column which contains the average error percentage (difference between the best lower bound and the current lower bound divided by the current lower bound and multiplied by 100).

Table 2. Makespan and Sum of Tardiness lower bounds comparison.

	LB|AGO|CT	LB|IR|CT	LB|IT~IR|CT	LB|AGOK|ST	LB|IT~IR|ST
RA/GD	0.35	0.03	0	1.68	2.07
RA/MD				0.31	0.87
DA/GD	5.35	0	0	30.1	0.03
DA/MD				15.5	0.07

In Table 2, for all instance sets, we can see that LB|IT~IR is always the best C_{max} lower bound followed by LB|IR and finally LB|AGO. We can observe that for ST, LB|AGOK is slightly better for regrouped arrivals, while LB|IT~IR is considerably better for dispersed arrivals. The CPU times remain always relatively small. The integer relaxation CPU times (~ 0.1 second) are only about three times greater than the other lower bounds.

In Table 3 we compare the approximation approaches with the lower bounds. The column GAP corresponds to the average error percentages between the best upper bound and the best lower bound associated to each instance. The performances are given separately for three criteria: CT, ST and ST*CT corresponding to hypotheses (B). The column BEST H contains the percentage of times a rapid approximation approach provides the best found value for (B). Only the name of the three best heuristic (HNDσ or HAσ, as defined in section 4) is given with the corresponding percentage.

Table 3. Upper and lower bounds comparison for C_{max} and the sum of weighted tardiness

	GAP CT	GAP ST	GAP(B)	BEST H		
RA/GD	0.84	15.2	16.2	HND_2 (27.9)	HND_4 (27.9)	HND_3 (14.2)
RA/MD		9.18	10.9	HND_2 (32.5)	HND_4 (32.5)	HA_2 (17.5)
DA/GD	3.71	31.8	39.5	HA_1 (15)	HA_3 (15)	HND_3 (13.8)
DA/MD		14.1	20.6	HA_4 (22.5)	HND_2 (20)	HND_4 (20)

For hypothesis (C) the comparison between the two lower bounds for the sum of weighted completions times is presented in Table 4. The performance is given in each column which contains the average error percentage. In average, LB|IT~IR is better for dispersed and for regrouped arrivals. LB|IT~IR computational times (~ 0.1 second) are six times more expensive than LB|AGOK in average.

Table 4. Sum of weighted completion times (ΣCT) lower bounds comparison.

	LB|AGOK| ΣCT	LB|IT~IR| ΣCT
RAD	0.73	0.51
DAD	16.9	0.04

As in Table 3, in Table 5 we compare the approximation approaches with the lower bounds for hypothesis (C). The performance is given for the sum of weighted completion times. The three best heuristics are given in column BEST H with the corresponding percentages.

Table 5. Upper and lower bounds comparison for the sum of weighted completion times

	GAP ΣCT		BEST H	
RAD	6.87	HND_2 (30)	HND_4 (30)	HA_2 (21.3)
DAD	15.3	HA_2 (24.4)	HA_4 (24.4)	HND_2 (13.5)

The evaluation of the performances of the approximation approaches compared with the lower bounds shows that the GAPs for the weighted tardiness (B) and weighted completion times (C), even by taking into account the resource constraints, using upper bounds of the idle times on intervals $[0,D_k]$, remain quite large. Two ways can decrease them, either to design even better lower bounds or to improve the approximation methods. The last one is certainly easier.

We implemented a branch and bound approach by using the elements introduced, some dominance properties and a classic branching scheme. In order to locate the optimal solution (OPT) in the interval [LB, UB], for each criterion, for each instance, we computed λ_{LB} = 100*(OPT-LB)/(UB-LB) and λ_{UB} = 100*(UB-OPT)/(UB-LB). Preliminary results show that, in average, for the three criteria, the optimal solution is closer to the best upper bound than to the best lower bound, independently of the instance family. For (A), average λ_{LB} = 54.4%. It is equal to 70.1% for (B) and to 76.4% for (C). We cannot give more detail of this work because of the limited space.

7 Conclusion and Future Work

We propose in this paper some new scheduling problems arising in the framework of supply chain nodes such as logistic platforms. It crosses two families of known scheduling problems that have not been studied conjointly, families with consumable resource arrivals and families with fixed dates for deliveries and even potentially a last flexible delivery date. We consider both maximal completion time and weighted sum of completion times criteria.

We present various lower bounds for the criteria and rapid approximation approaches to get upper bounds. These bounds are evaluated by a first set of experiments. It is to be noted that another bound was designed for the makespan criterion. The corresponding relaxation assumed that the arrivals are continuous and uniform over time. A polynomial algorithm proposed by Toker A. *et al.* 1991 for only one component and by Xie J. 1997 for several components is able to solve the obtained relaxed problem and consists in solving a two-machine flow shop problem with Johnson algorithm. Nevertheless, we do not present the corresponding experiments here because this lower bound is always the worst compared with the bounds presented in the paper.

We consider here only a single machine for the work inside the platform, some results can be very easily extended to identical parallel machines by using a global equivalent machine to get lower bounds and by placing the jobs on the first available machine for the upper bounds. Nevertheless, this will considerably increase the size of the branch and bound exploration and in this case it is better to explore powerful approximation methods such as meta-heuristics.

Other perspective consists to take into account more realistic hypotheses. In particular, it is possible that: some optimized tours are forbidden to some jobs; the truck capacity and the maximal number of trucks for each delivery are limited; trucks have compartments associated to families of products, whose capacity is limited; the number of visits to a given customer is limited and/or penalized… The capacity limitations can easily be taken into account in the integer relaxations and in the rapid approximation methods, while it is not at all obvious to integrate these new constraints in the agreeable order relaxations.

In conclusion, crossing scheduling with various supplier and customer delivery hypotheses provides a rich variety of NP-hard problems near to industrial reality and a challenge to practitioners.

References

Blazewics, J., Cellary, W., Slowinski, R., ad Weglarz, J.: Scheduling under resource constraints – deterministic models. Annals of Operations Research 7, J.C. Baltzer AG, Basel (1986)

Carlier, J.: Problèmes d'ordonnancement à contraintes de resources: Algorithmes et complexité. Doctorat d'état (1984)

Carlier, J., Moukrim, A., and Xu, H.: The project scheduling problem with production and consumption of resources: A list-scheduling based algorithm. Discrete Applied Mathematics 157(17), 3631—3642 (2009)

Cochand, M., de Werra, D., and Slowinski, R.: Preemptive scheduling with staircase and piecewise linear resource availability. Methos and Models of Operations Research 33, 297—313 (1989)

Gafarov, E.R., Lazarev, A.A., Werner, F.: Single machine scheduling with a non-renewable financial resource, http://www.math.uni-magdeburg.de/~werner/preprints/p10-07.pdf (2010)

Hall, N.G., Lesaoana, M., and Potts, C.N.: Scheduling with fixed delivery dates. Operations Research 49(1), 134—144 (2001)

Martello, S. and Toth, P.: Knapsack Problems. John Wiley & Sons, New York (1990)

Matsuo, H.: The weighted total tardiness problem with fixed shipping times and overtime utilization. Operations Research 36(2), 293—307 (1988)

Patterson, J.H., Slowinski, R., Talbot, F.B., and Weglarz, J.: An algorithm for a general class of precedence and resource constrained scheduling problems. In. Slowinski R and Weglarz, J., editor, Advances in Project Scheduling, pages 3—28. Elsevier Science Publishers B.V., Amsterdam (1989)

Slowinski, R.: Preemptive scheduling of independent jobs on parallel machines subject to financial constraints. European Journal of Operational Research 15, 366—373 (1984)

Toker, A., Kondakci, S., and Erkip, N.: Scheduling under a non-renewable resource constraint. Journal of the Operational Research Society 42(9), 911—814 (1991)

Xie, J.: Polynomial algorithms for a single machine scheduling problems with financial constraints. Operations Research Letters 21, 39—42 (1997)