Combinatorial Models for Multi-agent Scheduling Problems

Alessandro Agnetis[1], Dario Pacciarelli[2] and Andrea Pacifici[3]
[1]Università di Siena, [2]Dipartimento di Informatica e Automazione, Università di Roma, [3]Dipartimento di Ingegneria dell'Impresa, Università di Roma Italia

1. Abstract

Scheduling models deal with the best way of carrying out a set of jobs on given processing resources. Typically, the jobs belong to a single decision maker, who wants to find the most profitable way of organizing and exploiting available resources, and a single objective function is specified. If different objectives are present, there can be multiple objective functions, but still the models refer to a centralized framework, in which a single decision maker, given data on the jobs and the system, computes the best schedule for the whole system.
This approach does not apply to those situations in which the allocation process involves different subjects (agents), each having his/her own set of jobs, and there is no central authority who can solve possible conflicts in resource usage over time. In this case, the role of the model must be partially redefined, since rather than computing "optimal" solutions, the model is asked to provide useful elements for the negotiation process, which eventually leads to a stable and acceptable resource allocation.
Multi-agent scheduling models are dealt with by several distinct disciplines (besides optimization, we mention game theory, artificial intelligence etc), possibly indicated by different terms. We are not going to review the whole scope in detail, but rather we will concentrate on combinatorial models, and how they can be employed for the purpose on hand. We will consider two major mechanisms for generating schedules, auctions and bargaining models, corresponding to different information exchange scenarios.
Keywords: Scheduling, negotiation, combinatorial optimization, complexity, bargaining, games.

2. Introduction

In the classical approach to scheduling problems, all jobs conceptually belong to a single decision maker, who is obviously interested in arranging them in the most profitable (or less costly) way. This typically consists in optimizing a certain objective function. If more than one optimization criterion is present, the problem may become multi-criteria (see e.g. the thorough book by T'Kindt and Billaut [33]), but still decision problems and the corresponding solution algorithms are conceived in a centralized perspective.

This approach does not apply to situations in which, on the contrary, the allocation process involves different subjects *(agents),* each with its own set of jobs, requiring common resources, and there is no "superior" subject or authority who is in charge of solving conflicts on resource usage. In such cases, mathematical models can play the role of a negotiation support tool, conceived to help the agents to reach a mutually acceptable resource allocation. Optimization models are still important, but they must in general be integrated with other modeling tools, possibly derived from disciplines such as multi-agent systems, artificial intelligence or game theory.

In this chapter we want to present a number of modeling tools for multi-agent scheduling problems. Here we always consider situations in which the utility (or cost) function of the agents explicitly depends on some scheduling performance indices. Also, we do not consider situations in which the agents receiving an unfavorable allocation can be compensated through money. Scheduling problems with transferable utility are a special class of cooperative games called *sequencing games* (for a thorough survey on sequencing games, see Curiel et al. [9]). While interesting *per se,* sequencing games address different situations, in which, in particular, an initial schedule exists, and utility transfers among the agents take into account the (more or less privileged) starting position of each agent. This case does not cover all situations, though. For instance, an agent may be willing to complete its jobs on time as much as possible, but the monetary loss for late jobs can be difficult to quantify.

A key point in multi-agent scheduling situations concerns how information circulates among the agents. In many circumstances, the individual agents do not wish to disclose the details of their own jobs (such as the processing times, or even their own objectives), either to the other agents, or to an external coordinator. In this case, in order to reach an allocation, some form of structured protocol has to be used, typically an auction mechanism. On the basis of their private information, the agents bid for the common resource. Auctions for scheduling problems are reviewed in Section 3, and two meaningful examples are described in some detail. A different situation is when the agents are prone to disclose information concerning their own jobs, to openly bargain for the resource. This situation is better captured by *bargaining models* (Section 4), in which the agents must reach an agreement over a bargaining set consisting of all or a number of relevant schedules. In this context, two distinct problems arise. First, the bargaining set has to be computed, possibly in an efficient way.

Second, within the bargaining set it may be of interest to single out schedules which are compatible with certain assumptions on the agents' rationality and behavior, as well as social welfare. The computation of these schedules can also be viewed as a tool for an external facilitator who wishes to drive the negotiation process towards a schedule satisfying given requirements of fairness and efficiency. These problems lead to a new, special class of multicriteria scheduling problems, which can be called multi-agent or competitive scheduling problems. Finally, in Section 5, we present some preliminary results which refer to structured protocols other than the auctions. In this case, the agents submit their jobs to an external coordinator, who selects the next job for processing. In all cases, we review known results and point out venues for future research.

3. Motivation and notation

Multi-agent scheduling models arise in several applications. Here we briefly review some examples.

- Brewer and Plott [7] address a timetable design problem in which a central rail administration sells to private companies the right to use railroad tracks during given timeslots. Private companies behave as decentralized agents with conflicting objectives that compete for the usage of the railroad tracks through a competitive ascending-price auction. Each company has a set of trains to route through the network and a certain ideal timetable. Agent preferences are private values, but delayed timeslots have less value than ideal timeslots.
  Decentralized multi-agents scheduling models have been studied also for many other transportation problems, e.g., for aiport take-off and landing slot allocation problems [27]. For a comprehensive analysis of agent-based approaches to transport logistics, see [10].
- In [29, 4] the problem of integrating multimedia services for the standard SUMTS (Satellite-based Universal Mobile Telecommunication System) is considered. In this case the problem is to assign radio resources to various types of packets, including voice, web browsing, file transfer via ftp etc. Packet types correspond to agents, and have non-homogeneous objectives. For instance, the occasional loss of some voice-packet can be tolerated, but the packets delay must not exceed a certain maximum value, not to compromise the quality of the conversation. The transmission of a file via ftp requires that no packet is lost, while requirements on delays are soft.
- Multi-agent scheduling problems have been widely analyzed in the manufacturing context [30, 21, 32]. In this case the elements of the production process (machines, jobs, workers, tools...) may act as agents, each having its own objective (typically related to productivity maximization). Agents can also be implemented to represent physical aggregations of resources (e.g., the shop floor) or to encapsulate manufacturing activities (e.g., the planning function). In this case, using the autonomous agents paradigm is often motivated by the fact that it is too complex and expensive to have a single, centralized decision maker.
- Kubzin and Strusevich [16] address a maintenance planning problem in a two-machine shop. Here the maintenance periods are viewed as operations competing with the jobs for machines occupancy. An agent owns the jobs and aims to minimize the completion time of all jobs on all machines, while another agent owns the maintenance periods whose processing times are time dependent.

We next introduce some notation, valid throughout the chapter. A set of *m agents* is given, each owning a set of *jobs* to be processed on a single *machine.* The machine can process only one job at a time. We let *i* denote an agent, *i* = 1,..., *m,* $J^{(i)}$ its job set, and $J_j^{(i)}$ the *j-th* of its jobs, having *length* $p_j^{(i)}$. Let also $n_i = |J^{(i)}|$. Depending on specific situations, there are other quantities associated to each job, such as a *due date* $d_j^{(i)}$, a *weight* $w_j^{(i)}$, which can be regarded as a measure of the job's importance (for agent *i*), a *reward* R_j, which is obtained if the job is completed within its due date. We let J_q denote a generic job, when agent's ownship is immaterial. Jobs are all available from the beginning and once started, jobs cannot be preempted. A *schedule* is an assignment of starting times to the jobs. Hence, a

schedule is completely specified by the *sequence* in which the jobs are executed. Let σ be a schedule. We denote by $C_j^{(i)}(\sigma)$ the completion time of job $J_j^{(i)}$ in σ. If each agent owns exactly one job, we indicate the above quantities as $p_i,\ d_i,\ w_i,\ C_i(\sigma)$.
Agent *i* has a *utility function* $u^{(i)}(\sigma)$, which depends *exclusively* on the completion times of its own jobs. Function $u^{(i)}(\sigma)$ is nonincreasing as the completion times of its jobs grow. In some cases it will be more convenient to use a *cost function* $c^{(i)}(\sigma)$, obviously nondecreasing for increasing completion times of the agent's jobs.
Generally speaking, each agent aims at maximizing its own utility (or minimizing its costs). To pursue this goal, the agents have to make their decisions in an environment which is strongly characterized by the presence of the other agents, and will therefore have to carry out a suitable negotiation process. As a consequence, a decision support model must suitably represent the way in which the agents will interact to reach a mutually acceptable allocation. The next two chapters present in some detail two major modeling and procedural paradigms to address bargaining issues in a scheduling environment.

4. Auctions for decentralized scheduling

When dealing with decentralized scheduling methods, a key issue is how to reach a mutually acceptable allocation, complying with the fact that agents are not able (or willing) to exchange all the information they have. This has to do with the concept of private vs. public information. Agents are in general provided a certain amount of public information, but they will make their (bidding) decisions also on the basis of private information, which is not to be disclosed. Any method to reach a feasible schedule must therefore cope with the need of suitably representing and encoding public information, as well as other possible requirements, such as a reduced information exchange, and possibly yield "good" (from some individual and/or global viewpoint) allocations in reasonable computational time.
Actually, several *distributed scheduling* approaches have been proposed, making use of some degree of negotiation and/or bidding among job-agents and resource-agents. Among the best known contributions, we cite here Lin and Solberg [21]. Pinedo [25] gives a concise overview of these methods, see also Sabuncuoglu and Toptal [28]. These approaches are typically designed to address dynamic, distributed scheduling problems in complex, large-scale shop floor environments, for which a centralized computation of an overall "optimal" schedule may not be feasible due to communication and/or computation overhead. However, the conceptual framework is still that of a single subject (the system's owner) interested in driving the overall system performance towards a good result, disregarding jobs' ownship. In other words, in the context of distributed scheduling, market mechanisms are mainly a means to bypass technical and computational difficulties. Rather, we want to focus on formal models which explicitly address the fact that a limited number of agents, owning the jobs, bid for processing resources. In this respect, auction mechanisms display a number of positive features which make them natural candidates for complex, distributed allocation mechanisms, including scheduling situations. Auctions are usually simple to implement, and keep information exchange limited. The only information flow is in the format of bids (from the agents to the auctioneer) and prices (from the auctioneer to the agents). Also, the auction can be designed in a way that ensures certain properties of the final allocation.

Scheduling auctions regard the time as divided into time slots, which are the goods to be auctioned. The aim of the auction is to reach an allocation of time slots to the agents. This can be achieved by means of various, different auction mechanisms. Here we briefly review two examples of major auction types, namely an ascending auction and a combinatorial auction.
In this section we address the following situation. There is a set ***G*** of *goods,* consisting of ***T*** time slots on the machine. Processing of a job requires an integer number $p_j^{(i)}$ of time slots on the machine, which can, in turn, process only one job at a time. If a job $J_j^{(i)}$ is completed within slot $d_j^{(i)}$, agent *i* obtains a reward R_j. The agents bid for the time slots, and an auctioneer collects the bids and takes appropriate action to drive the bidding process towards a feasible (and hopefully, "good") allocation. We will suppose that each agent has a linear utility or value function (risk neutrality), which allows to compare the utility of different agents in monetary terms. The single-agent counterpart of the scheduling problem addressed here is the problem $1||\sum R_j U_j$.
What characterizes an auction mechanism is essentially how can the agents bid for the machine, and how the final allocation of time slots to the agents is reached.

4.1 Prices and equilibria

Wellman et al. [34] describe a scheduling economy in which the goods have *prices,* corresponding to amounts of money the agents have to spend to use such goods. An allocation is a partition of ***G*** into *i* subsets, $\boldsymbol{X} = \{\boldsymbol{X}_1, \mathbf{X}_2, \ldots, \boldsymbol{X}_m\}$. Let $v_i(X_i)$ be the value function of agent *i* if it gets the subset $X_i \subseteq G$ of goods. The *value* of an allocation $v(X)$ is the sum of all value functions,

$$v(X) = \sum_{i=1}^{m} v_i(X_i)$$

If slot *t* has price p_t, the *surplus* for agent *i* is represented by

$$v_i(X_i) - \sum_{t \in X_i} p_t$$

Clearly, each agent would like to maximize its surplus, i.e. to obtain the set $\boldsymbol{X}_i^*$ such that

$$H_i(p) = v_i(X_i^*) - \sum_{t \in X_i^*} p_t = \max_{S \subseteq G}\{v_i(S) - \sum_{t \in S} p_t\}$$

Now, if it happens that, for the current price vector *p,* each agent is assigned exactly the set $\boldsymbol{X}_i^*$, no agent has any interest in swapping or changing any of its goods with someone else's, and therefore the allocation is said to be in *equilibrium* for ***p***[1]. An allocation

[1] Actually, a more complete definition should include also the auctioneer, playing the role of the owner of the goods before they are auctioned. The value of good *t* to the auctioneer is $\boldsymbol{q}_t$, which is the starting price of each good, so that at the equilibrium $\boldsymbol{p}_t = \boldsymbol{q}_t$ for the goods which are not being allocated. For the sake of simplicity, we will not focus on the auctioneer and implicitly assume that $\boldsymbol{q}_t = 0$ for all *t*.

$\tilde{X} = \{\tilde{X}_1, \tilde{X}_2, \ldots, \tilde{X}_m\}$ is *optimal* if its total value is maximum among all feasible allocations.

Equilibrium (for some price vector ***p***) and optimality are closely related concepts. In fact, the following property is well-known (for any exchange economy):

Theorem 1: If an allocation ***X*** is in equilibrium at prices ***p***, then it is optimal.

In view of this (classical) result, one way to look at auctions is to analyze whether a certain auction mechanism may or may not lead to a price vector which supports equilibrium (and hence optimality). Actually, one may first question whether the converse of Theorem 1 holds, i.e., an optimal allocation is in equilibrium for some price vector. Wellman et al. show that in the special case in which all jobs are unit-length ($p_j^{(i)}$ = 1 for all $j \in J^{(i)}$, *i* = 1 , . . . , *m)* , an optimal allocation is supported by a price equilibrium (this is due to the fact that in this case each agent's preferences over time slots are additive, see Kelso and Crawford [15]). The rationale for this is quite simple. If jobs are unit-length, the different time slots are indeed independent goods in a market. No complementarities exist among goods, and the value of a good to an agent does not depend on whether the agent owns other goods. Instead, if one agent has one job of length $p_i = 2$, obtaining a single slot is worthless to the agent if it does not get at least another.

As a consequence, in the general case we cannot expect that *any* price formation mechanism reaches an equilibrium. Nonetheless, several auction mechanisms have been proposed and analyzed.

4.2 Interval scheduling

Before describing the auction mechanisms, let us briefly introduce an optimization subproblem which arises in many auction mechanisms.

Suppose that to use a certain time slot *t*, an agent *i* has to pay λ_t. Given the prices of the time slots, the problem is to select an appropriate subset of jobs from $J^{(i)}$ and schedule them in order to maximize the agent *i*'s revenue. Let u_{jt} the utility (given the current prices) of *starting* job $J_j^{(i)}$ at time *t*. Recalling that there is a reward R_j for timely completion of job $J_j^{(i)}$ (otherwise the agent may not have incentives to do any job), one has

$$u_{jt} = R_j^{(i)} \delta(d_j^{(i)} - t - p_j^{(i)} + 1) - \sum_{\tau=t}^{t+p_j^{(i)}-1} \lambda_\tau$$

where $\delta(x)$ = 1 if $x > 0$ and $\delta(x)$ = 0 otherwise. Letting $x_{jt} = 1$ if $J_j^{(i)}$ is started in slot *t*, we can formulate the problem as:

$$\max \ w = \sum_{j \in J^{(i)}} u_{jt} x_{jt} \qquad (1)$$

$$\sum_{j \in J^{(i)}} \sum_{\tau=t}^{t+p_j^{(i)}-1} x_{j\tau} \ \leq \ 1, t = 1, \ldots, T$$

$$\sum_{t=1}^{T} x_{jt} \leq 1, j \in J^{(i)}$$

Elendner [11] formulates a special case of (1) (in which $u_{jt} = u_t$ for all j) to model the winner determination problem in a sealed-bid combinatorial auction, and calls it Weighted Job Interval Scheduling Problem (WJISP), so we will also call it. In the next sections, we show that this problem arises from the agent's standpoint in several auction mechanisms. Problem (1) can be easily proved to be strongly NP-hard (reduction from 3-PARTITION).

4.3 Ascending auction

The ascending auction is perhaps the best known auction mechanism, and in fact it is widely implemented in several contexts. Goods are auctioned separately and in parallel. At any point in time, each good t has a current price β_t, which is the highest bid for t so far. The next bid for t will have to be at least $\beta_t + \epsilon$ (the *ask price).* Agents can asynchronously bid for any good in the market. When a certain amount of time elapses without any increase in a good's price, the good is allocated to the agent who bid last, for the current price.
This auction scheme leaves a certain amount of freedom to the agent to figure out the next bid, and in fact a large amount of literature is devoted to the ascending auction in a myriad of application contexts. In our context, we notice that a reasonable strategy for agent i is to ask for the subset $X^{(i)}$ maximizing its surplus for the current ask prices. This is precisely an instance of WJISP, which can therefore be nontrivial to solve exactly.
Even if, in the unit-length case, a price equilibrium does exist, a simple mechanism such as the ascending auction may fail to find one. However, Wellman et al. [34] show that the distance of the allocation provided by the auction from an equilibrium is bounded. In particular, suppose for simplicity that the number of agents m does not exceed the number of time slots. In the special case in which $|J^{(i)}| = 1$ and $p_i = 1$ for all i, the following results hold:
Theorem 2 *The final price of any good in an ascending auction differs from the respective equilibrium price by at most* $m\epsilon$.
Theorem 3 *The difference between the value of the allocation produced by an ascending auction and the optimal value is at most* $m^2\epsilon$.

4.4 Combinatorial mechanisms

Despite their simplicity, mechanisms as the ascending auction may fail to return satisfactory allocations, since they neglect the fact that each agent is indeed interested in getting *bundles* of (consecutive) time slots. For this reason, one can think of generalizing the concept of price equilibrium to combinatorial markets, and analyze the relationship between these concepts and optimal allocations. This means that now the goods in the market are no more simple slots, but rather slot intervals $[t_1, t_2]$. This means that rather than considering the price of single slots, one should consider prices of slot intervals. Wellman et al. show that it is still possible to suitably generalize the concept of equilibrium, but some properties which were valid in the single-slot case do not hold anymore. In particular, some problems which do not admit a price equilibrium in the single-unit case do admit an equilibrium in the larger space of combinatorial equilibria, but on the other hand, even if it exists, a combinatorial price equilibrium may not result in an optimal allocation.

In any case, the need arises for combinatorial auction protocols, and in fact a number has appeared in the literature so far. These mechanisms have in common the fact that through an iterative information exchange between the agents and the auctioneer, a compromise schedule emerges. The amount and type of information exchanged characterizes the various auction protocols. Here we review one of these mechanisms, adapting it from Kutanoglu and Wu [17][2]. The protocol works as follows.

1. The auctioneer declares the prices of each time slot, let λ_t, $t = 1,\ldots, T$ indicate the price of time slot *t.* On this basis, each agent *i* prepares a *bid* B_i, i.e., indicates a set of (disjoint) time slot intervals that the agent is willing to purchase for the current prices. Note that the bid is in the format of slot intervals, i.e. $B_i = \{[a_1^{(i)}, b_1^{(i)}], [a_2^{(i)}, b_2^{(i)}], \ldots\}$, meaning that it is worthless to the agent to get only a subset of each interval.
2. The auctioneer collects all the bids. If it turns out that no slot is required by more than one agent, the set of all bids defines a feasible schedule and the procedure stops. Else, a feasible schedule is computed which is "as close as possible" to the infeasible schedule defined by the bids.
3. The auctioneer modifies the prices of the time slots accounting for the *level of conflict* on each time slot, i.e., the number of agents that bid for that slot. The price modification scheme will tend to increase the price of the slots with a high level of conflict, while possibly decreasing the price of the slots which have not been required by anyone.
4. The auctioneer checks a stopping criterion. If it is met, the best solution (from a global standpoint) so far is taken as final allocation. Else, go back to step 1 and perform another round.

Note that this protocol requires that a bid consists of a number of disjoint intervals, and each of them produces a certain utility if the agent obtains it. In other words, we assume that it is not possible for the agent to declare preferences such as "either interval [2,4] or [3,5]". This scheme leaves a number of issues to be decided, upon which the performance of the method may heavily depend. In particular:

- How should each agent prepare its bid
- How should the prices be updated
- What stopping criterion should be used.

4.4.1 Bid preparation

The problem of the agent is again in the format of WJISP. Given the prices of the time slots, the problem is to select an appropriate subset of jobs from $J^{(i)}$ and schedule them in order to maximize the agent *i*'s revenue, with those prices. The schedule of the selected jobs defines the bid.

We note here that in the context of this combinatorial auction mechanism, solving (1) exactly may not be critical. In fact, the bid information is only used to update the slot prices, i.e., to figure out which are the most conflicting slots. Hence, a reasonable heuristic seems the most appropriate approach to address the agent's problem (1) in this type of combinatorial auctions.

[2] Unlike the original model by Kutanoglu and Wu, we consider here a single machine, agents owning multiple jobs, and having as objective the weighted number of tardy jobs.

4.4.2 Price update

Once the auctioneer has collected all agents' bids, it can compute how many agents actually request each slot. At the ***r***-th round of the auction, the level of conflict D_t^r of slot *t* is simply the number of agents requesting that slot, minus 1 (note that $D_t^r = -1$ if no agent is currently requesting slot ***t***). A simple rule to generate the new prices is to set them linearly in the level of conflict:

$$\lambda_t^{r+1} = \max\{0, \lambda_t^r + k^r D_t^r\}$$

where k^r is a step parameter which can vary during the algorithm. For instance, one can start with a higher value of k^r, and decrease it later on (this is called *adaptive tatonnement* by Kutanoglu and Wu).

4.4.3 Stopping criterion and feasibility restoration

This combinatorial auction mechanism may stop either when no conflicts are present in the union of all bids, or because a given number of iterations is reached. In the latter case, the auctioneer may be left with the problem of solving the residual resource conflicts when the auction process stops. This task can be easy if few conflicts still exist in the current solution. Hence, one technical issue is how to design the auction in a way that produces a good tradeoff between convergence speed and distance from feasibility. In this respect, and when the objective function is total tardiness, Kutanoglu and Wu [17] show that introducing price discrimination policies (i.e., the price of a slot may not be the same for all agents) may be of help, though the complexity of the agent subproblem may grow. As an example of a feasibility restoration heuristic, Jeong and Leon [18] (in the context of another type of auction-based scheduling system) propose to simply schedule all jobs in ascending order of their start times in the current infeasible schedule. Actually, when dealing with the multi-agent version of problem $1||\sum R_i U_i$, it may well be the case that a solution without conflicts is produced, since many jobs are already discarded by the agents when solving WJISP.

4.4.4 Relationship to Lagrangean relaxation

The whole idea of a combinatorial auction approach for scheduling has a strong relationship with Lagrange optimization. In fact, the need for an auction arises because the agents are either unwilling or unable to communicate all the relevant information concerning their jobs to a centralized supervisor. Actually, what makes things complicated is the obvious fact that the machine is able to process one job at a time only. If there were no such constraint, each agent could decide its own schedule simply disregarding the presence of the other agents. So, the prices play the role of multipliers corresponding to the capacity constraints.

To make things more precise, consider the problem of maximizing the overall total revenue. Since it is indeed a centralized problem, we can disregard agent's ownship. and simply use ***j*** to index the jobs. We can use the classical time-indexed formulation by Pritsker et al. [26][3]. The variable x_{jt} is equal to 1 if job ***j*** has started by time slot ***t*** and 0 otherwise. Hence, the revenue R_j is won by the agent if and only if job *j* has started by time slot $d_j - p_j + 1$.

[3] The following is a simplification of the development presented by Kutanoglu and Wu, who deal with job shop problems.

$$\begin{aligned}
\max \sum_{j\in J} R_j \; x_{j,d_j-p_j+1} & \\
x_{j,t+1} &\geq x_{jt} \quad j \in J, t = 1,\ldots,T \\
\sum_{j\in J}(x_{jt} - x_{j,t-p_j}) &\leq 1 \quad t = 1,\ldots,T \\
x_{jt} &\in \{0,1\} \quad j \in J, t = 1,\ldots,T
\end{aligned} \tag{2}$$

Constraints (2) express machine capacity. In fact, for each t there can be at most one job j which has already started at slot t and had not yet started at time $t - p_j$ (which means that j is currently under process in slot t) . Now, if we relax the machine capacity constraints in a Lagrangean fashion, we get the problem

$$L(\lambda) = \max \sum_{j\in J} R_j \; x_{j,d_j-p_j+1} \; - \; \sum_{t=1}^{T} \lambda_t \left(\sum_{j\in J}(x_{jt} - x_{j,t-p_j}) - 1 \right) \tag{3}$$
$$\begin{aligned}
x_{j,t+1} &\geq x_{jt} \quad j \in J, t = 1,\ldots,T \\
x_{jt} &\in \{0,1\} \quad j \in J, t = 1,\ldots,T
\end{aligned}$$

(Note that (3) can be solved by inspection, separately for each job.) The value $L(\lambda)$ is an upper bound on the optimal solution to (2). In an optimization context. one is typically interested in finding the best such bound, i.e..

$$L(\lambda^*) = \min_{\lambda} L(\lambda) \tag{4}$$

To solve (5), a very common approach is to iteratively update the multiplier vector λ by the subgradient algorithm, i.e., indicating by $\bar{x}$ the current optimal solution to (3) when $\lambda = \lambda^r$

$$\lambda^{r+1} = \lambda^r + s_r \left(\sum_{j\in J}(\bar{x}_{jt} - \bar{x}_{j,t-p_j}) - 1 \right) \tag{5}$$

where s_r is an appropriate step size. Now, observe that the term in braces in (5) is precisely what we previously called the level of conflict. Hence, it turns out that the subgradient algorithm is equivalent to a particular case of combinatorial auction (with adaptive tatonnement).

5. Bargaining problems and Pareto optimal schedules

We next want to analyze the scheduling problem from a different perspective. So far we supposed that it is possible, to a certain extent, to give a monetary evaluation of the quality of a solution. Actually, the value function of each agent might depend on certain schedule-related quantities which may not be easy to assess. For instance, completing a job beyond its due date may lead to some monetary loss, but also to other consequences (e.g. loss of customers' goodwill) which can be difficult to quantify exactly. In such cases, it appears more sensible that the agents directly negotiate upon possible schedules.

Bargaining models are a special class of cooperative games with non-transferable utility. For our scheduling situations, this means that the agents are, in principle, willing to disclose information concerning their jobs, and use this information to build a set of solutions and reach a satisfactory compromise schedule. Note that, unlike our previous assumptions, the agents may now have heterogeneous objectives. Also, for the sake of simplicity we deal here with the situation in which there are only two agents. However, the major concepts can be cast in a more general, *m*-agent, setting.

The viewpoint of axiomatic bargaining models is to characterize certain schedules, displaying some desirable properties which make them special candidates to be the outcome of negotiation. Here we want to apply some of these concepts to the scheduling setting, pointing out key issues from the modeling and computational viewpoints.

5.1 Bargaining problems

In a *bargaining problem,* two players (Agent 1 and Agent 2) have to negotiate a common strategy, i.e., choose an element of a set *S* of possible *agreements.* Each point $s \in S$ is a pair of payoffs for Agent 1 and 2 respectively, denoted by $u_1(s)$ and $u_2(s)$. If negotiation fails, Agents 1 and 2 get the payoff d_1 and d_2 respectively. A *bargaining problem* is a pair *(S, d),* where:

1. $S \subseteq \mathbb{R}^2$
2. $d = (d_1, d_2)$ is the *disagreement point,* i.e. the results of the failure of negotiation
3. at least one point $(u_1, u_2) \in S$ exists such that $u_1 > d_1$ and $u_2 > d_2$.

We next want to suitably characterize certain agreements in terms of efficiency and fairness. In fact, even if negotiation is helped by an external entity, it makes sense to select a few among all possible schedules, in order not to confuse the players with an excessive amount of information. A *solution* of a bargaining problem is an application φ which assigns to any problem instance (*S*, *d*) a subset of agreements (possibly, a single agreement) $\varphi(S, d) \subseteq S$. Consider now the following four axioms. which may or may not be satisfied by a certain solution φ:

1. **(Weak) Efficiency (PAR):**
   if $s \in \varphi(S, d)$, then there is no $t \in S$ such that $t_1 > S_1$ and $t_2 > S_2$
2. **Symmetry (SYM) :**
   if (*S*, *d*) is symmetric, $(u_1, u_2) \in \varphi(S, d)$ if and only if $(u_1, u_2) \in \varphi(S, d)$
3. **Scale Covariance (SC) :**
   $\forall \lambda_1, \lambda_2, \gamma_1, \gamma_2 \in \mathbb{R}$ such that $\lambda_1, \lambda_2 > 0$, if we let $S' = \{(\lambda_1 u_1 + \gamma_1, \lambda_2 u_2 + \gamma_2)$: $(u_1, u_2) \in S\}$ and $d' = (\lambda_1 d_1 + \gamma_1, \lambda_2 d_2 + \gamma_2)$, then $\varphi(S', d') = \{(\lambda_1 u_1 + \gamma_1, \lambda_2 u_2 + \gamma_2) | (u_1, u_2) \in \varphi(S, d)\}$
4. **Independence of Irrelevant Alternatives (HA) :**
   if we restrict the bargaining set to a subset S' such that $S' \cap \varphi(S) \neq \emptyset$, then $\varphi(S', d) = \varphi(S, d) \cup S'$.

The meaning of these axioms should be apparent. PAR means that if $s \in \varphi(S, d)$, then there is no other agreement such that both agents are better off, i.e., *s* is Pareto optimal. SYM implies that whenever the two agents have identical job sets and payoff functions, the outcome should give both players the same payoff. SC is related to the classical concept of utility, and states that the solution should not change if we use equivalent payoff representations. Finally, IIA says that the solution of a problem should not change if some agreements (not containing the solution) are removed from the bargaining set.

The classical notion of bargaining problem assumes *S* be a compact, convex subset of $\mathbb{R}^2$. For this case, Nash [23] proved that if and only if a solution φ (*S*, *d*) satisfies all four axioms, then φ (*S*, *d*) consists of a *single* agreement $\nu \in S$, given by:

$$\nu = \arg \max_{(u_1,u_2)\in S} [(u_1 - d_1)(u_2 - d_2)] \tag{6}$$

and ν is called the *Nash bargaining solution* (NBS). Since in our case the bargaining set is indeed a finite set of distinct schedules, the concept of NBS must be suitably extended. When *S* is a general, possibly discrete, set, Mariotti [22] showed that if and only if a solution $\varphi_N(S,d)$ satisfies all four axioms 1-4, then $\varphi_N(S,d)$ is given by

$$\varphi_N(S,d) = \{(u_1^*, u_2^*) \in S : (u_1^* - d_1)(u_2^* - d_2) = \max\,[(u_1 - d_1)(u_2 - d_2)]\} \tag{7}$$

The price we pay for this generalization is that $\varphi_N(S,d)$ may no longer consist of a single agreement. We still refer to set $\varphi_N(S,d)$ as the NBS.
So far we considered the payoffs (u_1, u_2) associated with an agreement. For our purpose, it is convenient to associate with each agreement a pair of *costs* (c_1,c_2), and let $\tilde{S}$ be the set of all cost pairs. Let now $\bar{c}_1$ and $\bar{c}_2$ be the costs of the *worst* agreements for Agent 1 and 2 respectively, i.e.

$$\begin{aligned} \bar{c}_1 &= \max\{c_1 : (c_1,c_2) \in \tilde{S}\} \\ \bar{c}_2 &= \max\{c_2 : (c_1,c_2) \in \tilde{S}\} \end{aligned} \tag{8}$$

In what follows, we assume that the players' costs in the event of breakdown are given by $\bar{c}_1$ and $\bar{c}_2$ respectively. This is equivalent to assuming that $\tilde{S}$ also includes the point $(\bar{c}_1, \bar{c}_2)$. Clearly, this models a situation in which the players are strongly encouraged to reach an agreement (other than $(\bar{c}_1, \bar{c}_2)$). Letting $u_1 = \bar{c}_1 - c_1$ and $u_2 = \bar{c}_2 - c_2$, we can define a bargaining problem *(S, d)* in which *S* is obtained from $\tilde{S}$ by a symmetry with respect to the point $(\bar{c}_1, \bar{c}_2)$, followed by a shift $(-\bar{c}_1, -\bar{c}_2)$, so that the disagreement point is the origin. In other words, we use as value function of a given agreement the *saving* with respect to the most costly alternative. The disagreement point is hence mapped in (0, 0) and the NBS is therefore given by

$$\begin{aligned} \varphi_N(S,(0,0)) &= \arg\max\{u_1 u_2 : (u_1,u_2) \in S\} \\ &= \arg\max\left\{(\bar{c}_1 - c_1)(\bar{c}_2 - c_2) : (c_1,c_2) \in \tilde{S}\right\} \end{aligned} \tag{9}$$

5.2 Application to scheduling problems

Let us now turn to our scheduling scenario. We denote the two players as Agent 1 (having job set $J^1 = \{J_1^1, J_2^1, \ldots, J_{n_1}^1\}$) and Agent 2 (with job set $J^2 = \{J_1^2, J_2^2, \ldots, J_{n_2}^2\}$). We call *1-jobs* and *2-jobs* the jobs of the two sets. The players have to agree upon a *schedule,* i.e., an assignment of starting times to all jobs. Agents 1 and 2 are willing to minimize cost functions $c^1(\sigma)$ and $c^2(\sigma)$ respectively, where σ denotes a schedule of the $n = n_1 + n_2$ jobs, and both cost functions are nondecreasing as each job's completion time increases. Note that we can restrict our analysis to *active* schedules, i.e., schedules in which each job starts immediately after the completion of the previous job. As a consequence, a schedule is completely specified by the *sequence* in which the jobs are scheduled. Also, we can indeed restrict our attention to Pareto optimal schedules only, since it does not appear reasonable that the agents ultimately agree on a situation from which penalizes both of them. In order to find Pareto-optimal schedules, consider the following problem:

```
{
$\Sigma := \emptyset$; $Q := +\infty$; $i := 0$
while the problem $1|f^2 \leq Q|f^1$ is feasible
        {
        $i := i + 1$
        $\sigma^{(i)}$ := Pareto-optimal solution of $1|f^2 \leq Q|f^1$
        $\Sigma := \Sigma \cup \sigma^{(i)}$
        $Q' := f^2(\sigma^{(i)})$
        $Q := Q' - \epsilon$
        }
}
```

Figure 1. Scheme for the enumeration of Pareto optimal schedules

$\text{PROBLEM } 1|c^2 \leq Q|c^1$. Given job sets J^1, J^2, cost functions $c^1(\cdot)$, $c^2(\cdot)$, and an integer Q, find σ^* such that

$$c^1(\sigma^*) = \min_{\sigma}\{c^1(\sigma)|c^2(\sigma) \leq Q\}.$$

Note that if σ^* is not Pareto optimal, a schedule of cost $c^1(\sigma^*)$ which is also Pareto optimal can be found by solving a logarithmic number of instances of $1|c^2 \leq \tilde{Q}|c^1$. In order to determine the whole set Σ of Pareto optimal schedules one can think of solving several instances of $1|c^2 \leq \tilde{Q}|c^1$, for decreasing values of Q (see Fig. 1).

A related problem is to minimize a convex combination of the two agents' cost functions [5]:

$\text{PROBLEM } 1||\lambda c^1 + (1-\lambda)c^2$. Given job sets J^1, J^2, cost functions $c^1(\cdot)$, $c^2(\cdot)$, and $\lambda \in [0,1]$, find a schedule σ^* such that $\lambda c^1(\sigma^*) + (1-\lambda)c^2(\sigma^*)$ is minimum.

The optimal solutions to $1||\lambda c^1 + (1-\lambda)c^2$, which are obtained for varying λ, are called *extreme* solutions. Clearly, all extreme solutions are also Pareto optimal, but not all Pareto optimal solutions are extreme. The following proposition holds.

Proposition 1: *If problem $1|c^2 \leq Q|c^1$ is solvable in time $O(g_1(n))$, and S has size $O(g_2(n))$, then $1||\lambda c^1 + (1-\lambda)c^2$ is solvable in time $O(g_1(n)g_2(n))$ for a given λ.*

Recalling (8) and (9), we can now formally define a scheduling bargaining problem. The bargaining set S consists of the origin d = (0, 0) plus the set of all pairs of payoffs $(u_1(\sigma), u_2(\sigma)) = (\bar{c}_1 - c_1(\sigma), \bar{c}_2 - c_2(\sigma))$, for $\sigma \in \Sigma$. The set of *Nash bargaining schedules* $\mathcal{N}$ is then

$$\mathcal{N} = \left\{\sigma^* : u_1(\sigma^*)u_2(\sigma^*) = \max_{\sigma\in\Sigma}\{u_1(\sigma)u_2(\sigma)\}\right\} \tag{10}$$

In order to analyze a scheduling bargaining problem, one is therefore left with the following questions:

- How hard is it to generate each point in S?

- How hard is it to generate extreme solutions in *S*?
- How large is the bargaining set *S*?
- How hard is it to compute the Nash bargaining solution?

The answers to these questions strongly depend on the particular cost functions of the two agents. Though far from drawing a complete picture, a number of results in the literature exist, outlining a new class of scheduling problems.
In view of (10), the problem of actually computing the set of Nash bargaining schedules is therefore a nonlinear optimization problem over a discrete set. In what follows, we study the computational complexity of generating the bargaining set *S,* for various cost functions *c*(·):

- (*maximum of regular functions*) $f_{\max}(\sigma) = \max_{j=1,\dots,n_i}\{f_j(C_j(\sigma))\}$, where each $f_j(\cdot)$ is nondecreasing in C_j .
- (*number of tardy jobs*) $\sum U_j(\sigma) = \sum_{j=1}^{n_i} U_j(\sigma)$, where $U_j(\sigma)$ = 1 if job J_j is late in σ and $U_j(\sigma)$ = 0 otherwise.
- (*total weighted flow time*) $\sum w_j C_j(\sigma) = \sum_{j=1}^{n_i} w_j C_j(\sigma)$.

We next analyze some of the scenarios obtained for various combinations of these cost functions.

5.3 $(f^1_{\max}, f^2_{\max})$

This case contains all cases in which each agent aims at minimizing the maximum of non-decreasing functions, each depending on the completion time of a job. Particular cases include makespan $C_{\max}$, maximum lateness $L_{\max}$, maximum tardiness $T_{\max}$ and so on.
The problem of finding an optimal solution to $1|f^2_{\max} \le Q|f^1_{\max}$ be efficiently solved by an easy reduction to the standard well-known, single-agent problem $1|prec|f_{\max}$, which can be solved, for example, with an $O(n^2)$ algorithm by Lawler [19]. Lawler 's algorithm for this special case may be sketched as follows. At each step, the algorithm selects, among unscheduled jobs, the job to be scheduled last. If we let $\bar{\tau}$ be the sum of the processing times of the unscheduled jobs, then any unscheduled 2-job J^2_k such that $f^2_k(\bar{\tau}) \le Q$ can be scheduled to end at $\bar{\tau}$. If there is no such 2-job, we schedule the 1-job J^1_h for which $f^1_h(\bar{\tau})$ is minimum. If, at a certain point in the algorithm, all 1-jobs have been scheduled and no 2-job can be scheduled last, the instance is not feasible. (We observe that the above algorithm can be easily extended to the case in which precedence constraints exist among jobs, even across the job sets J^1 and J^2. This may be the case, for instance, of assembly jobs that require components machined and released by the other agent.)
For each 2-job J^2_k, let us define a *deadline* D^2_k such that $f^2_k(C^2_k) \le Q$ for $C^2_k \le D^2_k$ and $f^2_k(C^2_k) > Q$ for $C^2_k > D^2_k$. The job set J^2 can be ordered a priori, in non-decreasing order of deadlines D^2_k., in time $O(n_2 \log n_2)$. At each step the only 2-job that needs to be considered is the unscheduled one with largest D^2_k. On the other hand, for each job in J^1, the corresponding $f^1_h(\bar{\tau})$ value must be computed. Supposing that each $f^1_h(\cdot)$ value can be computed in constant time, whenever no 2-job can be scheduled. all unscheduled 1-jobs may have to be tried out. Since this happens n_1 times, we may conclude with the following
Theorem 4: *Problem* $1|f^2_{\max} \le Q|f^1_{\max}$ *can be solved in time* $O(n_1^2 + n_2 \log n_2)$.
Using the above algorithm, we get an optimal solution σ^* to $1|f^2_{\max} \le Q|f^1_{\max}$. Let $Q_1 = f^1_{\max}(\sigma^*)$ and $Q_2 = f^2_{\max}(\sigma^*)$. In general, we are not guaranteed that σ^* is Pareto optimal. However, to find an optimal solution which is also Pareto optimal, we only need to

exchange the roles of the two agents, and solve an instance of $1|f^1_{\max} \leq Q^*|f^2_{\max}$ in which Q^* is the optimal value of $f^1_{\max}$ obtained with the Lawler's algorithm. Since this computation will require time $O(n_2^2 + n_1 \log n_1)$, we may state the following

Theorem 5: *A Pareto optimal solution to Problem $1|f^2_{\max} \leq Q|f^1_{\max}$ can be computed in time $O(n_1^2 + n_2^2)$.*

The set of all Pareto optimal solutions (i.e., the bargaining set *S)* can be found by the algorithm in Fig.l in which the quantity ϵ must be small enough in order not to miss any other Pareto-optimal solution. The ϵ to be used depends on the actual shape of the ***f*** functions. If their slope is small, small values of ϵ may be needed. Finally, in [1] it is shown that the following result holds.

Theorem 6: *There are at most n_1n_2 Pareto optimal schedules in $1||(f^1_{max}, f^2_{max})$.*

As a consequence, and recalling Proposition 1, the problem $1|\lambda f^1_{\max} + (1-\lambda)f^2_{\max}$ can be solved in time $O(n_1^3 n_2 + n_1 n_2^3)$ for any value of $\lambda \in$ [0, 1]. Similarly, from Theorem 6, finding the Nash bargaining solution simply requires to compute values $u_1(\sigma)u_2(\sigma)$ in equation (10) for all possible pairs of Pareto optimal solutions, which can be done in time $O(n_1^3 n_2 + n_1 n_2^3)$.

5.4 $(\sum_j C^1_j, f^2_{\max})$

This case contains all cases in which Agent 1 aims at minimizing the completion time of its jobs, while Agent 2 wants to minimize the maximum of nondecreasing functions, each depending on the completion time of the jobs in ***J***^{2}.

5.4.1 $1|f^2_{\max} \leq Q|\sum_j C^1_j$

In this section we show that $1|f^2_{\max} \leq Q|\sum_j C^1_j$ is polynomially solvable. Two lemmas allow us to devise the solution algorithm for this problem.

Lemma 1: *Consider a feasible instance of $1|f^2_{\max} \leq Q|\sum_j C^1_j$ and let $\tau = P_1 + P_2$. If there is a 2-job J^2_k such that $f^2_k(\tau) \leq Q$, then there is an optimal schedule in which J^2_k is scheduled last, and there is no optimal schedule in which a* 1*-job is scheduled last.*

Proof. Let σ' be an optimal schedule in which J|| is not scheduled last, and let σ^* be the schedule obtained by moving J^2_k in the last position. For any job J^X_i other than J^2_k, $C^X_i(\sigma^*) \leq C^X_i(\sigma')$ and therefore, $\sum_j {C_j}^1(\sigma^*) \leq \sum_j {C_j}^1(\sigma')$. In particular, if a 1-job is last in σ', then $\sum_j {C_j}^1(\sigma^*) \leq \sum_j {C_j}^1(\sigma')$, thus contradicting the optimality of σ'. For what concerns J^2_k, its completion time is now τ, and by hypothesis $f^2_k(\tau) \leq Q$. Hence, due to the regularity of $f^2_k(\cdot)$ for all *k*, the schedule σ^* is still feasible and optimal.

The second lemma specifies the order in which the 1-jobs must be scheduled.

Lemma 2: *Consider a feasible instance of $1|f^2_{\max} \leq Q|\sum_j C^1_j$ and let $\tau = P_1 + P_2$. If for all 1-jobs J^2_k, $f^2_k(\tau) > Q$, then in any optimal schedule a longest l-job is scheduled last.*

Proof. The result is established by a simple interchange argument.

The solution algorithm is similar to the one in Section 5.3. At each step, the algorithm selects a job to be scheduled last among unscheduled jobs. If possible, a 2-job is selected. Else, the longest l-job is scheduled last. If all 1-jobs have been scheduled and no 2-job can be scheduled last, the instance is infeasible. It is also easy to show that the complexity of this algorithm is dominated by the ordering phase, so that the following result holds.

Theorem 7: $1|f^2_{\max} \le Q|\sum_j C^1_j$ *can be solved in time* $\boldsymbol{O}(n_1 \log n_1 + n_2 \log n_2)$.

The optimal solution obtained by the above algorithm may not be Pareto optimal. The next lemma specifies the structure of any optimal solution to $1|f^2_{\max} \le Q|\sum_j C^1_j$ thus including the Pareto optimal ones. Given a feasible sequence σ, in what follows we define 2-*block* a maximal set of consecutive 2-jobs in σ.

Lemma 3: *Given a feasible instance of* $1|f^2_{\max} \le Q|\sum_j C^1_j$*, for all optimal solutions:*

(1) The partition of 2-jobs into 2-blocks is the same

(2) The 2-blocks are scheduled in the same time intervals.

Proof. See [1].

Lemma 3 completely characterizes the structure of the optimal solutions. The completion times of the 1-jobs are the same in all optimal solutions, modulo permutations of identical jobs. The 2-blocks are also the same in all optimal solutions, the only difference being the internal scheduling of each 2-block. Hence, to get a Pareto optimal schedule, it is sufficient to order the 2-jobs in each 2-block with the Lawler's algorithm [19]. Notice that selecting at each step the 2-job according to the Lawler's algorithm implies an explicit computation of the $f^2_k(\cdot)$ functions. As a result, we cannot order the 2-jobs a priori, and the following theorem holds.

Theorem 8: *An optimal solution to* $1|f^2_{\max} \le Q|\sum_j C^1_j$ *which is also Pareto optimal can be computed in time* $O(n_1 \log n_1 + n_2{}^2)$.

We next address the problem of determining the size of the bargaining set. From Lemma 2 we know that in any Pareto optimal schedule, the jobs of $\boldsymbol{J}^1$ are SPT-ordered. As Q decreases, the optimal schedule for $1|f^2_{\max} \le Q|\sum_j C^1_j$ changes. It is possible to prove [1] that when the constraint on the objective function of agent 2 becomes tighter, the completion time of no 1-job can decrease. As a consequence. once a 2-job *overtakes* (i.e. it is done before) a 1-job in a Pareto optimal solution. As Q is decreased, no reverse overtake can occur when Q decreases further. Hence, the following result holds.

Theorem 9: *There are at most* $n_1 n_2$ *Pareto optimal schedules in* $1||(\sum_j C^1_j, f^2_{max})$.

Finally, in view of Proposition 1 and Theorem 8, one has that an optimal solution to $1|\lambda \sum_i C^1_i + (1-\lambda) f^2_{\max}$, as well as the Nash bargaining solution can be found in time.

5.5 $(\sum_j w_j C^1_j, f^2_{\max})$

This case contains all cases in which Agent 1 aims at minimizing the weighted completion time of his/her jobs, while Agent 2 wants to minimize the maximum of nondecreasing functions, each depending on the completion time of the jobs in $\boldsymbol{J}^2$. The complexity of the weighted problem is different from the unweighted cases of previous section. For this reason we address this case separately from the unweighted one.

5.5.1 $1|f^2_{\max} \le Q|\sum w_j C^1_j$

We next address the weighted case of problem $1|f^2_{\max} \le Q|\sum w_j C^1_j$. A key result for the unweighted case, shown in Lemma 2 is that 1-jobs are SPT ordered in all optimal solutions, which would be also the optimal solution for the single agent problem $1||\sum_j C^1_j$. The optimal solution for the single agent problem $1||\sum_j w_j C_j^1$ can be computed with the Smith's rule, i.e., ranking the jobs by nondecreasing values of the ratios $\frac{p_j}{w_j}$. The question then arises of whether 1-jobs are processed in this order also in the Pareto optimal solutions of $1|f^2_{\max} \leq Q|\sum w_j C_j^1$. Unfortunately, it is easy to show that this is not the case in general. Consider the following example.

Example 1: *Suppose that set **J**² consists of a single job J_1^2 having processing time p_1^2* = 10, *and that $f^2_{\max} = C^2_{\max}$, i.e., that Agent 2 is only interested in competing his/her job within time Q* = 20. *Agent 1 owns four jobs $J_1^1, J_2^1, J_3^1, J_4^1$ with processing times and weights shown in table 1. Sequencing the 1-jobs with the Smith's rule and then inserting the only 2-job in the latest feasible position, one obtains the sequence $\sigma = \{J_1^1, J_1^2, J_2^1, J_3^1, J_4^1\}$, with $f^1(\sigma)$* = 9*6+7*21+4*24+5* 28 = 437, *while the optimal solution is $\sigma^* = \{J_1^1, J_4^1, J_1^2, J_2^1, J_3^1\}$, with $f^1(\sigma^*)$* = 9*6+5*10+7*25+4*28= 391.

job	p_j	w_j
J_1^1	6	9
J_2^1	5	7
J_3^1	3	4
J_4^1	4	5

Table 1. Data for Agent 1 in Example 1

We note that in the optimal solution of Example 1, Consecutive jobs of Agent 1 are *WSPT*-ordered. Yet it is not trivial to decide how to insert the 2-jobs in the schedule. Indeed even when there is only one job of Agent 2 and its objective is to minimize $f^2_{\max} = C^2_{\max}$, the problem turns out to be binary NP-hard. The reduction uses the well-known NP-hard KNAPSACK problem.

KNAPSACK. Given two sets of nonnegative integers {u_1, u_2, $\cdots$, u_n} and {w_1, w_2, $\cdots$, w_n}, and two integers b and W, find a subset $S \subseteq$ {1,... ,n} such that $\sum_{i \in S} u_i \leq b$ and $\sum_{i \in S} w_i$ is maximum.

Theorem 10: $1|f^2_{\max} \leq Q|\sum w_j C_j^1$ *is binary NP-hard.*

Proof. We give a sketch of the proof, details can be found in [1]. Given an instance of KNAPSACK, we define an instance of $1|f^2_{\max} \leq Q|\sum w_j C_j^1$ as follows. Agent 1 has n jobs, having processing times $p_i^1 = u_i$ and weights $w_i^1 = w_i$, i = 1,..., n. Agent 2 has only one *very long* job, having processing time B. Also, we set the deadline for the 2-job to b+B. Now, the completion times of all the 1-jobs ending after the 2-job will pay B. If B is very large, the best thing one can do is therefore to maximize the *total weight* of the 1-jobs scheduled before the 2-job. Since these 1-jobs have to be scheduled in the interval [0,b], this is precisely equivalent to solving the original instance of KNAPSACK.

5.5.2 Generating extreme solutions

Interestingly, while $1|f^2_{\max} \leq Q|\sum w_j C_j^1$ is NP-hard, the corresponding Problem $1||\lambda \sum w_j C_j^1 + (1-\lambda) C^2_{\max}$ can be solved in polynomial time, as observed by Smith and Baker [5]. First note that, in any Pareto optimal solution, with no loss of generality Agent 2 may process its jobs consecutively, since it is only interested in its last job's completion time. Hence, we may replace the 2-jobs with a single (block) job for Agent 2. The processing time of the block job equals the sum of the processing times of the 2-jobs. Consider now $1||\lambda \sum w_j C_j^1 + (1-\lambda) C_{\max}^2$. This problem is now equivalent to the classical (single- agent) $1|| \sum w_j C_j$ with n+1 jobs where Agent 1 jobs have weights λw_j, j = 1, . . . , n, while the weight of the single 2-job is 1 — λ. By applying the Smith's rule we may solve the problem in time ***O***(n log n). Moreover. note that, varying the values of λ, the position of the 2-job changes in the schedule. while the 1-jobs remain in *WSPT* order. In conclusion, by repeatedly applying the above described procedure we are able to efficiently generate $O(n_1)$ extreme Pareto optimal solutions.

5.5.3 Generating the bargaining set

Despite the fact that the number of extreme solutions is polynomial, Pareto optimal solutions are not polynomially many.

Lemma 4: *Consider an instance $1|f_{\max}^2 \le Q| \sum w_j C_j^1$ in which Agent 2 has a single job of unit length, while Agent 1 has n_1 jobs. For each 1-job i (i $\in$ {1, 2, . . . , n_1} = J^1), $p_i^1 = w_i^1 = 2^{i-1}$. Then, for every active schedule, the quantity $C_1^2 + \sum_{i=1}^{n_1} w_i C_i^1$ is constant and equal to $\frac{1}{3}(1 + 2^{2n_1+1})$.*

Proof. Given any active schedule σ, consider two adjacent 1-jobs ***j*** and ***k***. Let t be the starting time of job ***j*** and t+p_j the starting time of job ***k***. The contribution to the objective function of the two jobs is then $w_j(t + p_j) + w_k(t + p_j + p_k)$. Consider now the schedule $\bar{\sigma}$ in which the two jobs are switched: the contribution of the two jobs to the objective function is now $w_k(t + p_k) + w_j(t + p_k + p_j)$. Observe now that $w_j\, p_k$ = $w_k\, p_j$ for any pair of jobs in $J^1 \cup J^2$ (since w_i = p_i for each job), thus proving that σ and $\bar{\sigma}$ have the same value of the objective function. This implies that any active schedule produces the same value of the quantity $C_1^2 + \sum_{i=1}^{n_1} w_i C_i^1$. This value can be computed, for example, by considering the sequence: $J_1^2, J_1^1, J_2^1, \ldots, J_{n_1}^1$. We have:

$$C_1^2 + \sum_{i=1}^{n_1} w_i C_i^1 = 1 + \sum_{i=1}^{n_1} 2^{i-1} 2^i = 1 + \sum_{i=1}^{n_1} 2^{2i-1} = 1 + \sum_{i=1}^{2n_1} 2^i - \sum_{i=1}^{n_1} 2^{2i}.$$

Since $\sum_{i=1}^{n_1} 2^{2i} = 2 \sum_{i=1}^{n_1} 2^{2i-1}$ we can write: $\sum_{i=1}^{n_1} 2^{2i-1} = \sum_{i=1}^{2n_1} 2^i - 2 \sum_{i=1}^{n_1} 2^{2i-1}$. Hence, we obtain: $\sum_{i=1}^{n_1} 2^{2i-1} = \frac{1}{3}(\sum_{i=1}^{2n_1} 2^i) = \frac{1}{3}(2^{2n_1+1} - 2)$. In conclusion, the quantity $C_1^2 + \sum_{i=1}^{n_1} w_i C_i^1$ is equal to $\frac{1}{3}(1 + 2^{2n_1+1})$, and the thesis follows.

Theorem 11: $1 | C_{\max}^2 \le Q| \sum w_j C_j^1$ *has an exponential number of Pareto optimal solutions.*

In order to prove that the instance of Lemma 4 has an exponential number of Pareto optimal pairs, consider that, for any value $1 \le Q \le 2^{n_1}$, there is a subset of ***J***^{1} whose total length equals $Q - 1$. This implies that there is a feasible solution to $1 | C_{\max}^2 \le Q| \sum w_j C_j^1$ where $C_{\max}^2 = Q$ and $\sum w_j C_j^1 = \frac{1}{3}(1 + 2^{2n_1+1}) - Q$. This is clearly a Pareto optimal solution and therefore we have 2^{n_1} Pareto optimal solutions.

Finally, we observe that no polynomial algorithm is known for finding a Nash bargaining solution in the set of all Pareto optimal solutions and the complexity of this problem is still open.

5.6 Other scenarios

In the cases considered above, we observed that when problem $1|f^2 \leq Q|f^1$ is polynomially solvable, the number of Pareto optimal solutions is polynomially bounded, whereas if the same problem is NP-hard there are exponentially (pseudo-polynomially) many Pareto optima. Nonetheless, no general relationship links these two aspects. As an example, consider Problem $1|\sum U_j^2 \leq Q|\sum w_j C_j^1$ which is NP-hard (this is a trivial consequence of Theorem 10). Clearly, the number of Pareto optimal solutions of any problem of the class $1|\sum U_j^2 \leq Q|f^1$ cannot exceed n_2, for any possible choice of Agent 1 objective.

Table 2 summarizes the complexity results of several two-agent scheduling problems. In particular, note that the complexity of $1||\lambda \sum C_j^1 + (1-\lambda) \sum U_j^2$ is not known yet. In [24] it is shown that $1|\sum U_j^2 \leq Q|\sum C_j^1$ is NP-hard under high multiplicity encoding (see also [6]), which does not rule out the possibility of a polynomial algorithm for the general case. If this problem were polynomially solvable, this would imply that $1||\lambda \sum C_j^1 + (1-\lambda) \sum U_j^2$ is in P.

In [3], some extensions of the results reported in Table 2 to the case of *k* agents are addressed. When multiple agents want to minimize $f_{\max}$ objective functions, a simple separability procedure enables to solve an equivalent problem instance with a reduced number of agents. In Table 3 we report the complexity results for some maximal polinomially solvable cases.

(f^1, f^2)	*Complexity of* $1\|f^2 \leq Q\|f^1$	*Size of the bargaining set* Σ	*Complexity of* $1\|\|\lambda f^1 + (1-\lambda) f^2$
$(f_{\max}^1, f_{\max}^2)$	$O(n_1^2 + n_2 \log n_2)$ [1]	$O(n_1 n_2)$ [1]	$O(n^3 \log n)$
$(\sum w_j C_j^1, C_{\max}^2)$	NP-hard [1]	exp. [3]	$O(n \log n)$[5]
$(\sum w_j C_j^1, L_{\max}^2)$	NP-hard [1]	exp. [3]	NP-hard [5]
$(\sum C_j^1, f_{\max}^2)$	$O(n \log n)$ [1]	$O(n_1 n_2)$ [1]	$O(n^3 \log n)$
$(\sum U_j^1, f_{\max}^2)$	$O(n \log n)$ [1]	$O(n_1)$ [1]	$O(n^2 \log n)$
$(\sum U_j^1, \sum U_j^2)$	$O(n^3)$ [1]	$\min\{n_1, n_2\}$ [1]	$O(n^4)$
$(\sum w_j U_j^1, \sum w_j U_j^2)$	NP-hard [20, 14]	Open	NP-hard [20, 14]
$(\sum C_j^1, \sum U_j^2)$	Open*	$O(n_2)$ [1]	Open
$(\sum w_j C_j^1, \sum U_j^2)$	NP-hard [1]	$O(n_2)$ [1]	NP-hard [5]
$(\sum C_j^1, \sum C_j^2)$	NP-hard [1]	exp. [1]	$O(n \log n)$ [5]
$(\sum w_j C_j^1, \sum w_j C_j^2)$	NP-hard [1]	exp. [1]	$O(n \log n)$ [5]

*The problem is NP-Hard under high multiplicity encoding [24]

Table 2. Summary of complexity results for decision and Pareto optimization problems for two-agent scheduling problems

$(f^1, \ldots, f^k)$	*Complexity of* $1\|f^1 \leq Q_1, \ldots, f^k \leq Q_k\|\cdot$	*Complexity of* $1\|\|\sum_{i=1}^k \lambda_i f^i$
$(f_{\max}^1, \ldots, f_{\max}^k)$	$O(n \log n)$ [3]	$O(n \log n)$
$(\sum C_j^1, f_{\max}^2, \ldots, f_{\max}^k)$	$O(n \log n)$ [3]	$O(n \log n)$
$(\sum U_j^1, \ldots, \sum U_j^k)$	$O(\sum_{h=1}^k n_h \prod_{j=1}^k n_j)$ [3]	$O(n^{2k})$
$(\sum U_j^1, \ldots, \sum U_j^p, f_{\max}^{p+1}, \ldots, f_{\max}^k)$	$O(\sum_{h=1}^p n_h \prod_{j=1}^p n_j + n \log n)$ [3]	$O(n^{2k})$

Table 3. Maximal polynomially solvable cases of multi-agent scheduling problems

6. Single resource scheduling with an arbitrator

In this section we briefly describe a different decentralized scheduling architecture, making use of a special coordination protocol. A preliminary version of the results presented here are reported in [2]. The framework considered is the same described in Section 5.2. Again, we consider a single machine (deterministic, non-preemptive) scheduling setting with two agents, owning job sets $\boldsymbol{J}^1 = \{A_1, \dots A_{n(A)}\}$ and $\boldsymbol{J}^2 = \{B_1, \dots B_{n(B)}\}$ respectively. Each agent wants to optimize its own cost function.

In this scenario, the agents are not willing to disclose complete information concerning their jobs to the other agent, but only to an external coordination subject, called *arbitrator.* The protocol consists of iteratively performing the following steps:

1. Each agent submits one job to the arbitrator for possible processing.
2. The arbitrator selects one of the submitted jobs, according to a priority rule $\mathcal{R}$, and schedules it at the end of the current schedule. We assume the current schedule is initially empty.

The priority rule is public information, whereas jobs characteristics are private information of the respective agent. The information disclosed by the agent concerns the processing time and/or other quantities relevant to apply the priority rule. After the job is selected and scheduled, its processing time is communicated also to the other agent.

Let $f^h : \{C_i^h : i \in J^h\} \to \mathbb{R}_+$, be the cost function Agent *h, h* = 1,2, wants to minimize. We next report some results concerning the following cases for $f^h(\cdot)$:

1. total completion time $\sum_{i=1} C_i^h$;
2. total weighted completion time $\sum_{i=1} w_i C_i^h$; and
3. number of late jobs $\sum_{i=1} U_i^h$.

As for the arbitrator rules $\mathcal{R}$, we consider

1. Priority rules *SPT, WSPT,* and *EDD* if the arbitrator selects the next job to be scheduled between the two candidates according their minimum processing time, weighted processing time, and due date, respectively.
2. Round-Robin rule *RR:* if agents' jobs are alternated.
3. *k*-$\mathcal{R}$*:* a hybrid rule where at most *k* consecutive jobs of the same agents are selected according to rule $\mathcal{R}$.

In the following, we indicate the problem where the agents want to minimize cost functions $f^1(\cdot)$ and $f^2(\cdot)$ and the arbitrator rule is $\mathcal{R}$, as $1|\mathcal{R}|f^1, f^2$.

Example 2: *Consider the two job sets in Table 4- Suppose the arbitrator has a rule R = EDD. Then, it will choose the earliest due date job between the two candidates. If the job is late in the sequence it is cancelled from the schedule. The resulting sequence is illustrated in Table 5.*

1-jobs	1	2	3
d_i^1	3	6	12
p_i^1	2	2	5

2-jobs	1	2	3
d_i^2	5	7	20
p_i^2	3	2	9

Table 4. Job sets of Example 2

On this basis, one is interested in investigating several scenarios, for different objective functions f^1, f^2 and arbitrator rules $\mathcal{R}$. In particular, it is of interest to analyze the deviation of the resulting sequence from some "social welfare" solution (whatever the definition of such solution is). Of course, in an unconstrained scenario, one agent, say Agent 1, could improve its objective function penalizing the objective of Agent 2 *and* the global performance. This would obviously occur if Agent 1 were free to decide the schedule of its jobs. As a consequence, one may ask if arbitrator's rules exist that make a *fair* behavior convenient for both agents.

In the remainder of this section, we are addressing the latter problem in different scenarios, assuming as a social welfare solution one which minimizes the (unweighted) sum of the cost functions of the two agents.

Definition 1: *Given objective functions $f^1(\cdot)$ and $f^2(\cdot)$ for the two agents, a* global optimum *is a sequence of $J^1 \cup J^2$ minimizing the sum of the two objectives $f^1 + f^2$.*

Definition 2: *Given objective functions $f^1(\cdot)$ and $f^2(\cdot)$ for the two agents, and the priority rule $\mathcal{R}$, an $\mathcal{R}$* -optimum *is a sequence of $J^1 \cup J^2$ minimizing the sum of the two objectives $f^1 + f^2$, among all the sequences which can be obtained applying rule $\mathcal{R}$.*

Early jobs of σ	A_1	B_1	B_2	A_3
Start. time	0	3	5	7
Compl. time	3	5	7	12

Table 5. Resulting sequence of Example 2

Definition 3: *Given the objective function $f^h(\cdot)$ of Agent h (h* = 1,2*), and the priority rule $\mathcal{R}$, a h-*optimum *is a sequence of jobs in $J^1 \cup J^2$ that minimizes f^h, among all the sequences which can be obtained applying rule $\mathcal{R}$.*

6.1 WSPT rule

We start our analysis with $1|WSPT|\sum_i w_i^1 C_i^1, \sum w_i^2 C_i^2$, that is the problem where both agents want to minimize their total (weighted) completion times and the arbitrator selects the next job choosing the one with the smallest processing time over weight ratio. Hereafter this ratio is referred to as *density* δ_i. For a job *i*, with processing time p_i and positive weight w_i, $\delta_i = \frac{p_i}{w_i}$. In classical single machine scheduling, a sequence of jobs in non-decreasing order of density *(WSPT-order)* minimizes $\sum_i w_i C_i$. By standard pairwise interchange arguments, it is easy to prove the following:

Proposition 4: *In the scenario $1|WSPT|\sum_i w_i^1 C_i^1, \sum w_i^2 C_i^2$, if both agents propose WSPT-ordered candidate jobs, the resulting sequence is 1- optimal, 2-optimal, $\mathcal{R}$ -optimal and globally optimal.*

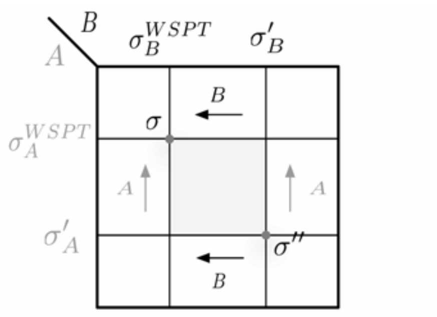

Figure 2. Schedule σ is a Nash Equilibrium

Note that if we view the scenario $1|WSPT|\sum_i w_i^1 C_i^1, \sum w_i^2 C_i^2$ as a game in which each agent's strategy is the order in which the jobs are submitted, Proposition 4 can be equivalently stated saying that the pair of strategies consisting in ordering the jobs in WSPT is the only Nash equilibrium.

6.2 Round Robin rules

We call *round-robin schedule* a schedule where the jobs of the two agents are alternating. In this section, we deal with the problems arising when the arbitrator selects the candidate jobs according to a rule $\mathcal{R}$ = *RR*, i.e., the only feasible schedules are round-robin schedules. Note that this embodies a very simple notion of fairness: the agents take turns in using the resource.

For simplicity, in the following we assume an equal number of jobs $n_1 = n_2 = n$ for the two agents. With no loss of generality, we also suppose that each agent's jobs are numbered by nondecreasing length, $p_i^h \leq p_{i+1}^h$, for all $i = 1,..., n-1$ and $h = 1,2$.

6.2.1 $1|RR|\sum_i C_i^1,\ \sum_i C_i^2$

When the agents want to minimize their total completion times among all possible round-robin rules, their strategy simply consists in presenting their jobs in *SPT*-order. Again by standard pairwise interchange arguments, one can show that the following propositions holds.

Proposition 5: *In the scenario* $1|RR|\sum_i C_i^1,\ \sum_i C_i^2$, *if both agents propose SPT-ordered candidate jobs, the resulting sequence is 1-optimal, 2-optimal and RR-optimal.*

Let σ_{RR}^* be the *RR*-optimal schedule. Since it may not be globally optimal, we want to investigate the *competitive ratio of* σ_{RR}^*, i.e., the largest possible ratio between the cost of the RR-optimal schedule σ_{RR}^* and the optimal value for an unconstrained schedule of the same jobs.

Let us denote with

- *OPT*$_{SPT}$ the cost of a global optimum of $1||\sum_{i \in J^1 \cup J^2} C_i$
- c_{SPT} *(A)* the cost of an optimal solution of $1||\sum_{i \in J^1} C_i$
  c_{SPT} *(B)* the cost of an optimal solution of $1||\sum_{i \in J^1} C_i$
- *OPT*$_{RR}$*:* the total cost of σ_{RR}^*.

The following proposition holds.

Proposition 6: $OPT_{RR} \leq 2OPT_{SPT}$

Proof sketch. It suffices to note that $OPT_{RR} \leq 2c_{SPT}(A) + 2c_{SPT}(B) \leq 2OPT_{SPT}$.

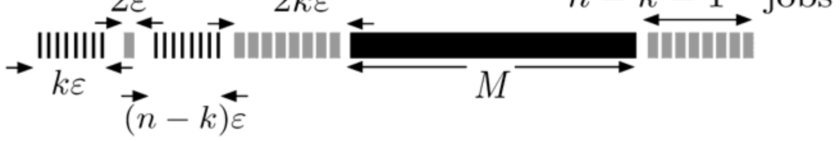

Figure 3. Istance with *O(n)* competitive ratio. The case with *n - k - 1 < k* is depicted

6.2.2 $1|k\text{-}\mathcal{R}|\sum_i C_i^1,\ \sum_i C_i^2$

Hereafter, we consider a generalization of the round-robin rule. The arbitrator selects one between the two candidate jobs according rule $\mathcal{R}$, but *no more* than k consecutive jobs of the same agent are allowed in the final sequence. We call *k round-robin* (briefly, k-$\mathcal{RR}$) a schedule produced by the above rule. (Note that for $k = 1$ we reobtain round-robin schedules.)

Let us denote by $\sigma^*_{k\text{-}\mathcal{R}}$ a k-$\mathcal{R}$-optimal schedule. One may show very easily that

Proposition 7: *In the scenario* $1|\ k\text{-}\mathcal{R}|\sum_i C_i^1, \sum C_i^2$; *if both agents propose SPT-ordered candidate jobs, the resulting sequence is 1- optimal, 2- optimal and* $k - \mathcal{RR}$*-optimal.*

However, unlike the round-robin case, the competitive ratio

$$\frac{\text{cost}(\sigma^*_{k\text{-}\mathcal{R}})}{OPT_{SPT}}$$

for general values of k may be arbitrarily bad. The example illustrated in Figure 3, for sufficiently large values of M and small enough positive ε, has a $O(n)$ ratio.

6.3 EDD rules

We conclude this section with an example in which $\mathcal{R}$ = *EDD*, i.e., the arbitrator schedules the most urgent between the two submitted. It is interesting to note as this rule may produce arbitrarily bad sequences in scenario $1|EDD|\sum_i U_i^1,\ \sum_i U_i^2$.

Consider the instance of $1|EDD|\sum_i U_i^1,\ \sum_i U_i^2$ reported in Table 6, where $n >> m >> 1$. Figure 4 illustrates a global optimum. Note that there are $m + 1$ tardy jobs for Agent 1 and 0 for Agent 2.

1-jobs	1	2	3	...	$m+2$	$m+3$
p^i	1	$2n$	3	...	3	2
d^i	2	2	$2n+4$	...	$2n+3m+1$	$2n+3m+4$
	A^1	A^2	A^3	...	A^{m+2}	A^{last}

2-jobs	1	2	...	$n+1$	$n+2$	...	$m+n+1$	$m+n+2$
p^i	1	2	...	2	3	...	3	1
d^i	1	$2n+2$	...	$2n+2$	$2n+5$	...	$2n+3m+2$	$2n+3m+5$
	B^1	B	...	B	B'	...	B'	B^{last}

Table 6. An instance of $1|EDD|\sum_i U_i^1,\ \sum_i U_i^2$

A 1-optimal schedule is obtained if Agent 1 just *skips* A^1 in the list of candidate jobs, and submits all the others, from 2 to $m + 3$. The resulting schedule is illustrated in Figure 5: in this case there are $n + m + 1$ tardy jobs and just one job of Agent 1 is tardy. Hence, again the competitive ratio is $O(n)$. In such situations, the coordination rule turns out to be ineffective and a preliminary negotiation between the two agents seems a much more recommendable strategy to obtain agreeable solutions for the two agents.

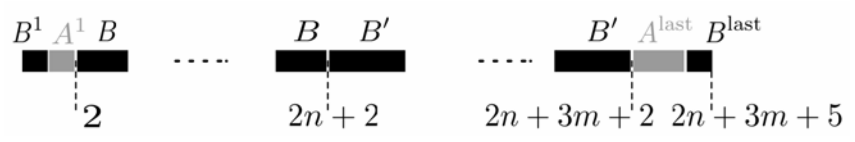

Figure 4. Globally optimal schedule for the instance of Table 6

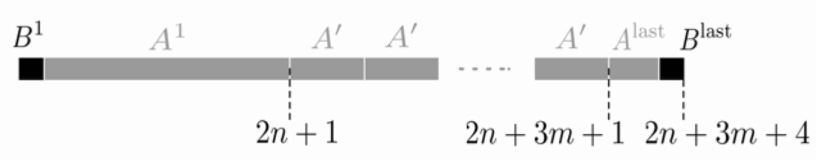

Figure 5. 1-optimal schedule for the instance of Table 6

7. Conclusions

In this chapter we have described a number of models which are useful when several agents have to negotiate processing resources on the basis of their scheduling performance. Research in this area appears at a fairly initial stage. Among the topics for future research we can mention:

- An experimental comparison of different auction mechanisms for scheduling problems, in terms of possibly addressing general systems (shops, parallel machines...)
- Analyzing several optimization problems, related to finding "good" overall solutions to multi-agent scheduling problems
- Designing and analyzing effective scheduling protocols and the corresponding agents' strategies.

8. References

Agnetis, A., Mirchandani, P.B., Pacciarelli, D., Pacifici, A. (2004), Scheduling problems with two competing agents, *Operations Research,* 52 (2), 229-242. [1]

Agnetis, A., Pacciarelli, D., Pacifici, A. (2006), Scheduling with Cheating Agents, Communication at AIRO 2006 Conference, Sept. 12-15, 2006. Cesena. Italy. [2]

Agnetis, A., D. Pacciarelli, A. Pacifici (2007), Multi-agent single machine scheduling, *Annals of Operations Research,* 150, 3-15. [3]

Arbib, C., S. Smriglio, and M. Servilio. (2004). A Competitive Scheduling Problem and its Relevance to UMTS Channel Assignment. *Networks,* 44 (2), 132-141. [4]

Baker, K., Smith C.J. (2003), A multi-objective scheduling model, *Journal of Scheduling,* 6 (1),7-16. [5]

Brauner, N., Y. Crama, A. Grigoriev, J. van de Klundert (2007), Multiplicity and complexity issues in contemporary production scheduling. *Statistica Neerlandica* 61 1, 7591. [6]

Brewer, P.J., Plott, C.R. (1996), A Binary Conflict Ascending Price (BICAP) Mechanism for the Decentralized Allocation of the Right to Use Railroad Tracks. *International Journal of Industrial Organization,* 14, 857-886. [7]

Brucker, P. (1998), *Scheduling Algorithms,* Springer-Verlag, Heidelberg. [8]

Curiel, I., Pederzoli, G., Tijs, S. (1989), Sequencing games, *European Journal of Operational Research ,* 40, 344-351. [9]

Davidsson, P., L. Henesey, L. Ramstedt, J Tornquist, F. Wernstedt (2005), An analysis of agent-based approaches to transport logistics, *Transportation Research Part C,* 13 255-271. [10]

Elendner, T. (2003) Scheduling adn combinatorial auctions: Lagrangean relaxation-based bounds for the WJISP, Institut fur Betriebswirtschaftslehre. University of Kiel, working paper n.570. [11]

French, S., (1986), *Decision Theory - An Introduction to the Mathematics of Rationality,* North-Holland. [12]

Garey, M.R., Johnson, D.S. (1979), *Computers and Intractability,* Freeman and Company, New York. [13]

Karp, R.M. (1972), Reducibility among combinatorial problems, In Complexity of computer computations (*Proc. Sympos., IBM Thomas J. Watson Res. Center,* Yorktown Heights, N.Y., 1972), Plenum, New York, 85-103. [14]

Kelso, A.S., Crawford, V.P. (1982), Job matching, coalition formation, and gross substitutes, *Econometrica*, 50, 1483-1504. [15]

Kubzin, M.A., V.A. Strusevich (2006), Planning Machine Maintenance in Two-Machine Shop Scheduling, *Operations Research*, 54 (4), 789-800. [16]

Kutanoglu, E., Wu, D. (1999), On combinatorial auction and Lagrangean relaxation for ditributed resource scheduling, *IIE Transactions*, 31**,** 813-826. [17]

Jeong, I.-J., Leon, V.J. (2005), A single-machine distributed scheduling methodology using cooperative interaction via coupling agents, *IIE Transactions*, 37. 137-152. [18]

Lawler, E.L. (1973), Optimal sequencing of a single machine subject to precedence constraints, *Management Science*, 19, 544-546. [19]

Lawler, E.L., J. M. Moore (1969), A Functional Equation and Its Application to Resource Allocation and Sequencing Problems *Management Science* 16, 1. Theory Series, 77-84.q[20]

Lin G.Y., Solberg J.J. (1992), Integrated shop floor control using autonomous agents, *HE Transactions*, 24 (3), 57-71. [21]

Mariotti, M. (1998) Nash bargaining theory when the number of alternatives can be finite. *Social choice and welfare*, 15, 1998, 413-421. [22]

J.F. Nash. The Bargaining Problem. Econometrica, 18, 1950, 155-162. [23]

Ng, C.T., T.C.E. Cheng, J.J. Yuan (2006), A note on the complexity of the problem of two-agent scheduling on a single machine *Journal of Combinatorial Optimization*, 12, 387-394. [24]

Pinedo, M., Scheduling: theory, algorithms and systems, 2nd edition, Prentice-Hall, 2001. [25]

Pritsker, A., Watters, L., Wolfe, P. (1969), Multiproject scheduling with limited resources: a zero-one programming approach, *Management Science: Theory*, 16(1), 93-108. [26]

Rassenti, S.J., V.L. Smith, R.L. Bulfin (1982), A combinatorial mechanism for airport time slot allocation, *Bell Journal of Economics*, 13 402-417. [27]

Sabuncuoglu, I., Toptal, A. (1999), Distributed scheduling, I: A review of concepts and applications, *Technical paper* IEOR 9910, department of Indutrial Engineering, Bilkent University, Ankara, Turkey. [28]

Schultz, D., Seoung-Hoon Oh, C. F. Grecas, M. Albani, J. Sanchez, C. Arbib, V. Arvia, M. Servilio, F. Del Sorbo, A. Giralda, G. Lombardi(2002), A QoS Concept for Packet Oriented S-UMTS Services, *Proceedings of the 1ST MOBILE SUMMIT* 2002, Thessaloniki (Greece). [29]

Shen, W., Q. Hao, H. Joong Yoon, D.H. Norrie (2006), Applications of agent-based systems in intelligent manufacturing: An updated review, *Advanced Engineering Informatics,* 20 (4), 415-431. [30]

Smith, W.E. (1956), Various Optimizers for Single Stage Production, *Naval research Logistics Quarterly*, 3, 59-66. [31]

Sousa P., Ramos C. (1999), A distributed architecture and negotiation protocol for scheduling in manufacturing systems, *Computers in Industry*, 38 (2), 103-113. [32]

T'Kindt, V., Billaut, J.C. (2002), Multicriteria Scheduling, Springer Verlag, Heidelberg. [33]

Wellman, M.P., W.E. Walsh, P.R. Wurman, J.K. MacKie-Mason (2001), Auction Protocols for Decentralized Scheduling, *Games and Economic Behavior*, 35 (1/2), 271-303. [34]