Online Mechanism Design for Scheduling Non-Preemptive Jobs under Uncertain Supply and Demand

Philipp Ströhle*, Enrico H. Gerding◇, Mathijs M. de Weerdt○,
Sebastian Stein◇, Valentin Robu◇

* Karlsruhe Institute of Technology, Germany, philipp.stroehle@kit.edu
◇ University of Southampton, UK, {eg,ss2,vr2}@ecs.soton.ac.uk
○ Delft University of Technology, Netherlands, m.m.deweerdt@tudelft.nl

ABSTRACT

We design new algorithms for the problem of allocating uncertain, flexible, and multi-unit demand online given uncertain supply, in order to maximise social welfare. The algorithms can be seen as extensions of the *expectation* and *consensus* algorithms from the domain of online scheduling. The problem is especially relevant to the future smart grid, where uncertain output from renewable generators and conventional supply need to be integrated and matched to flexible, non-preemptive demand. To deal with uncertain supply and demand, the algorithms generate multiple scenarios which can then be solved offline. Furthermore, we use a novel method of reweighting the scenarios based on their likelihood whenever new information about supply becomes available. An additional improvement allows the selection of *multiple* non-preemptive jobs at the same time. Finally, our main contribution is a novel online mechanism based on these extensions, where it is in the agents' best interest to truthfully reveal their preferences. The experimental evaluation of the extended algorithms and different variants of the mechanism show that both achieve more than 85% of the offline optimal economic efficiency. Importantly, the mechanism yields comparable efficiency, while, in contrast to the algorithms, it allows for strategic agents.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Scheduling; G.3 [**Probability and statistics**]: Probabilistic algorithms; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

Keywords

Online Mechanism Design, Scheduling, Uncertainty

1. INTRODUCTION

The availability of electrical energy from renewable sources such as wind and solar has been increasing rapidly in the past years, and is expected to significantly increase even further in the near future.[1] However, electricity generation from many renewable sources cannot be easily controlled and is often difficult to predict accurately. Therefore, balancing demand and supply in the power system becomes increasingly challenging when relying on the ramping capabilities of conventional supply only. Alternatively, this problem can be addressed by introducing more flexibility on the demand side and allowing loads to be deferred. Thus, a major challenge in future energy systems is to schedule flexible loads online given both uncertain future demand as well as uncertain supply of renewable energy.

To meet this challenge, we introduce and evaluate several novel algorithms for the online scheduling of deferrable loads, that take into account probabilistic information about future supply and demand. Furthermore, we use a mechanism design approach to incentivise agents on the demand side to be truthful about their flexibility and the value of the loads. Specifically, we consider the problem of scheduling multiple non-preemptive loads (i.e., loads that, once started, cannot be interrupted) with a fixed load profile (i.e., the power transfer rate of a load is a function of time from the starting point of the load) and uncertain supply from relatively cheap or even free renewable energy.

Our algorithms extend existing single machine scheduling algorithms called *expectation* [1] and *consensus* [4], which take uncertainty about future jobs into account. These algorithms generate scenarios from a probability distribution, and the scheduling problem is solved for each of these scenarios using an offline scheduling algorithm. The former approach uses the scenarios to estimate the expected utility of scheduling a certain job first, whereas in the latter, scenarios vote to decide which load to schedule first. We extend both algorithms for settings with variable, uncertain supply where multiple, heterogeneous jobs, can run simultaneously. Furthermore, we use the concept of precommitment to make these algorithms into truthful mechanisms.

Recently, *consensus* was applied in a smart grid setting with a specific focus on the context of energy allocation for the charging of electric vehicles [12]. While that work addresses the problem of uncertain future demand, the model is restricted to deterministic future energy supply with constant marginal cost. In this paper, we specifically focus on the problem of uncertain supply. More precisely, we as-

[1] In Germany, 22.9% of electricity was supplied from renewable sources in 2012 [11], up from 12% in 2006, while in the UK the government has committed to ensure that 30% of energy supply comes from renewable sources by 2020 [3].

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

sume that there are two sources of electricity: renewable supply, which is uncertain but free, and conventional generation, which is always available but costly. Another difference to [12] is that, in their work, loads are presumed to be preemptive, which helps to reduce the complexity of the problem. However, in many real life settings jobs requiring electricity are non-preemptive (i.e., they cannot be easily interrupted and restarted).[2] For this reason, in this paper we focus on non-preemptive loads.

Our work is also related to [13], who consider the problem of scheduling deferrable loads. However, there are some important differences with the model and algorithms they consider. First of all, although they also introduce a predictive approach, they use a point prediction of renewable supply. The disadvantage is that their approach only considers the expected supply, and does not take into consideration (auto-)correlations over time, which is common in renewable supply (e.g., there can be days with high supply, and ones with low supply, but the expected value might be very unlikely in practice). In contrast, our *consensus* based approach considers several scenarios which are sampled from the distribution, and correlation is taken into account when generating such scenarios. Another important difference is that they assume loads to be preemptive, as requiring a certain total amount of energy, and as being characterised by a flexible power consumption rate. In contrast, we assume non-preemptive loads with a fixed load profile and require that started loads need to be completed. Finally, unlike [13] and similar to [12], we assume that loads have monetary values, and our aim is to maximise the difference between the value of allocated loads and the cost of using non-renewable energy, i.e., to maximise social welfare.

Recently, researchers in the multi-agent community have begun looking at adapting online scheduling heuristics to deal with strategic agents. Such agents may misreport the value, arrival time or deadline for their jobs if, through such a misreport, they can get a better allocation or pay less. In order to ensure that an online scheduling heuristic is truthfully implementable with strategic agents a key criterion to be satisfied is *monotonicity* [8]: if a job has a type that is better in any of its dimensions than another (e.g., higher value, lower consumption rate, shorter length, earlier arrival or later deadline), and no worse in any other dimension, then its allocation must not be worse.

There are two main approaches to ensuring monotonicity of online allocations. One approach involves using of *output ironing* [7, 2], or cancelling that part of the allocation that breaks monotonicity constraints. While this is a principled approach, often a large part of the final allocation may need to be cancelled, and computing the ironing decisions can be intractable in realistically-sized settings. Another approach, which we also use in this paper, is to partially precommit to allocate to agents of sufficiently high value in the future, irrespective of future arrivals [12]. While this approach may slightly reduce efficiency, because in some settings it imposes additional constraints on future schedules, it has the advantage that it prevents strategic agents from misreporting.

Specifically, our main contributions are as follows:

- We consider, for the first time, the problem of online scheduling non-preemptive jobs and uncertain supply of resources.
- We present and compare several variants of two new algorithms for this setting: an extension of the *consensus* approach [4] and an extension of the *expectation* approach [1] to deal with selecting *multiple* jobs at each time step and variable supply.
- We use mechanism design to produce truthful variants of these algorithms by using the concept of precommitment.

The setup of this paper is as follows. First, we formalise the online scheduling problem. Then, we explain how we extend both *consensus* and the *expectation*-based approach to deal with possibly selecting multiple jobs at each time step, followed by a description of the issues and solutions to make these methods incentive compatible. We conclude with an experimental validation and a discussion.

2. PROBLEM FORMULATION

Our online scheduling problem with non-preemptive loads is characterised by demands of different values and requirements, which arrive online, and supply from two sources: an uncertain future amount of low-cost power from renewables, and costly (conventional) generation. The decision of the scheduler concerns which of the incoming loads to schedule, and when, in order to maximise expected social welfare (defined below). We note that even the offline version of the problem (with perfect knowledge of demand and supply) is of combinatorial complexity, as finding the optimal schedule requires a combinatorial number of subsets to be evaluated and compared [4]. Hence, approximation algorithms are needed. Before we present the algorithms, we detail the formal definition of demand, supply, and the schedule.

2.1 Demand

We consider a setting where jobs arrive over a fixed finite time horizon (e.g., a day), modelled by a set of discrete time steps $T = \{1, \dots, \mathcal{T}\}$. A *job* $j \in J$, where J is the set of all jobs, is characterised by a type $\langle v_j, r_j, l_j, a_j, d_j \rangle$, which comprises its value $v_j \in \mathbb{R}^+$, consumption rate $r_j \in \mathbb{R}^+$ (the amount of supply needed per time step), job length $l_j \in T$, arrival time $a_j \in T$, and departure time or deadline $d_j \in T$. Given this, we denote the total amount of energy required to serve a job as $q_j = l_j \cdot r_j$. A job j cannot start before a_j, must end before d_j, and is non-preemptive, i.e., once it is started, it must continue running for l_j time steps (which we assume to be bounded by a constant maximum length). Thus, the latest start time for a job is $d_j - l_j$, and the flexibility is its difference with a_j, i.e., $d_j - l_j - a_j$. We assume that the set of jobs J is not known a-priori, but is only revealed online as jobs arrive in the system. However, the scheduler has a probability distribution of future jobs and their properties, such as their valuation and length.

2.2 Supply

Supply is available from two sources, renewable and conventional, that exhibit different properties. Renewable sources, such as wind or solar power, are characterised by negligible marginal cost but also uncertain availability. For simplicity, we assume that costs for renewable energy are zero in this paper (although the algorithms carry over to settings where

[2]Examples include washing machines, in a domestic setting, and a variety of heavy duty electrical machinery in an industrial setting.

the marginal cost is constant, and can easily be generalised to other cost functions). Furthermore, we assume that the maximum renewable power available is given by a stochastic process $X_T = (X_1, \ldots, X_\mathcal{T})$, whose realisation $x_t \sim X_t$ only becomes known at t. Importantly, the supply can be correlated over time, which means that realisations at time t provide information about future supply (as explained in Section 5.1, in our experiments we model this stochastic process using a hidden Markov model). For example, if there is wind in the morning, it is more likely that there will be wind during the following hours.

Conventional generation, on the other hand, is characterised by unlimited output and a deterministic cost function c which is non-decreasing in the amount of power supplied at time t. In particular, in this paper, we take these costs to be described by a linear function (constant marginal cost), i.e., $c_c(p_c) = b \cdot p_c, \quad b > 0$. Supply is perishable, i.e., electrical energy that is not immediately consumed cannot be stored and consumed in the future.

2.3 Schedule

The solution to the online scheduling problem is a *schedule* $s = \langle s_1, s_2, \ldots, s_\mathcal{T} \rangle$, which defines for every time step t a set of jobs $s_t \subseteq J$ to start at that time. A *feasible* schedule s should meet the following constraints, for all times $t \in T$ and for all jobs j started at time t, i.e., $j \in s_t$:

- j cannot start before its arrival, i.e., $t \geq a_j$,
- j must finish by its deadline, i.e., $t \leq d_j - l_j$,
- j can be started at most once, i.e., $\forall t, t' \in T$: if $j \in s_{t'}$ and $j \in s_t$ then $t = t'$.

We use $s = \langle\rangle$ to denote the empty schedule, i.e., where $s_t = \emptyset$ for all $t \in T$. Furthermore, given a schedule s and a time t, we denote the set of *running jobs* by $R_t(s) = \{j \mid j \in s_{t'}, t' \leq t, t' + l_j > t\}$. The net profit (or social welfare) $w(s)$ of a schedule s is then defined by the value of all scheduled jobs minus the cost,

$$w(s) = \sum_{t \in T} \left(\sum_{j \in s_t} v_j - c\Big(\max\Big\{ \sum_{j \in R_t(s)} r_j - x_t, 0 \Big\} \Big) \right) \quad (1)$$

which is the value we aim to maximise. To illustrate the problem, consider the following example.

EXAMPLE 1. *Consider a setting with two time steps t_1 and t_2 and two agents $\{a_1, a_2\}$, each requiring exactly one unit of energy. The value of a_1's job is $v_1 = 7$, and it can run during either t_1 or t_2, while a_2 has a job which has value $v_2 = 5$, but only during t_1. At t_1 one unit of renewable supply is available with associated costs $c = 0$. For t_2, the mechanism* expects *one unit of renewable supply to be available, but there is a very small chance that this is not realised and then the alternative from conventional generation will be very expensive, i.e., $c = 10$. Moreover, the mechanism expects, with high probability, no further arrivals, but there is a small chance agent a_3 with a high value ($7 < v_3 < 10$) will enter the market at t_2. At t_1 the expected optimal allocation is to allocate a_2 (who has an earlier deadline), and postpone a_1. Now assume that a_3 enters the market at t_2. This leads to a_1 being discarded at t_2, and a_3 being allocated. With hindsight (i.e., offline optimal), however, it would have been better to discard a_2, allocate a_1 at t_1, and a_3 at t_2.*

Algorithm: OFFLINE (J, x, s, t)
$J' \leftarrow \{j \in J \mid j \notin s, d_j \geq t + l_j\}$
for $j \in$ SORT(J') `// by decreasing value density`
do
  $t_{min}, c_{min} \leftarrow$ COSTMINIMALSTARTTIME(j, x, s, t)
  if $c_{min} < v_j$ **then**
    $s_{t_{min}} \leftarrow s_{t_{min}} \cup \{j\}$
return s

Algorithm 1: Greedy offline scheduler.

2.4 Strategic behaviour

When designing the algorithms, we also need to consider strategic behaviour on the demand side. Since the types of the jobs constitute private information, we would like to incentivise agents to reveal their types truthfully. Otherwise, agents could speculate and the scheduler might take suboptimal decisions based on incorrect/manipulated information. Specifically, the aim is to design a mechanism, i.e., a scheduling algorithm and corresponding payments, which is *dominant-strategy incentive compatible*, i.e., reporting truthfully maximises an agent's utility, regardless of the behaviour of other agents. To guarantee this property, we assume that each job is owned by a different agent. Additionally, we guarantee *individual rationality*, which means that the payment is never more than the job's value, and is zero if a job is not run. In Section 4 we return to these issues in detail, focusing first on the online scheduling problem.

3. MODEL-BASED ONLINE SCHEDULING

In this section, we present our extensions of the online algorithms from [4]. Similar to that work, our algorithms deal with uncertainty by sampling multiple future scenarios using an appropriate model of the system (in our case, sampled realisations of the future supply of renewable energy). Then, at each time step, an offline algorithm is used to solve each of these scenarios, and the resulting schedules are combined to yield the best decision to take in the current time step. Unlike previous work, however, our algorithms are able to schedule multiple jobs per time step (rather than a single one), deal with uncertain future supply (rather than assuming this to be deterministic), and incorporate costs of exceeding the available supply (by using conventional energy).

As our algorithms rely on solving instances of an offline version of the scheduling problem, we first detail an algorithm for this in Section 3.1. Then, we provide a generic online algorithm that all our approaches follow, and conclude this section with two novel online scheduling algorithms: m-CONSENSUS, and m-EXPECTATION.

3.1 Offline Scheduling Algorithm

In the offline variant of our scheduling problem, we assume that all jobs J and the realisation of the supply x are known in advance. However, finding an optimal schedule even in this case is known to be computationally hard, as it is a generalisation of the NP-hard parallel machine scheduling problem [9]. For this reason, we use a greedy scheduling heuristic and refer to this as OFFLINE (Algorithm 1).

This algorithm has two more arguments, s and t, which are used by our online algorithms later to encode past (and fixed) scheduling decisions (s) as well as the current time (t), which is the earliest time at which new jobs may be

scheduled.[3] The algorithm first sorts the available jobs J by decreasing value density v_j/q_j. For each job in this order, the function CostMinimalStartTime then computes the starting time $t_{min} \in T, t_{min} \geq t$, that minimises the additional cost incurred by adding job j to s_t. If the associated minimum cost, c_{min}, is less than the value of the job, it is included in the schedule s. The scheduler thus effectively performs two operations for each job, deciding whether to start it and if so, at what time. For n jobs, the run time of this scheduler is $\mathcal{O}(n \log n)$ for sorting, plus $\mathcal{O}(n\mathcal{T})$ for finding the best start times for all jobs, so bounded by $\mathcal{O}(n\mathcal{T})$ because $\mathcal{T}$ is typically much larger than $\log n$.

3.2 Online Scheduling Algorithms

In online settings, new information is revealed over time, requiring sequential decision making. We consider two algorithms that are executed in the following context.

First, a set of $\mathcal{N}$ scenarios is created. Each scenario $i \in \{1, .., \mathcal{N}\}$ consists of the tuple $\langle J^i, x^i \rangle$, where J^i is a randomly sampled realisation of future *demand* which we refer to as the set of "virtual jobs", and x^i is a sampled realisation of *supply* based on a probabilistic supply model (or historical data).

Then, at every time point t, one of the two online algorithms is invoked with the following arguments: the currently startable jobs (not scheduled so far) J_r, the realisation of the renewable supply x up until and including t, as well as the schedule so far. The scenarios include future demand and supply, i.e., $J^i(t) = \{j | j \in J^i \text{ where } a_j \geq t+1\}$ and $x^i(t) = \langle x_t, x^i_{t+1}, x^i_{t+2}, \ldots, x^i_T \rangle$. Each scenario i is assigned a weight $\mathcal{L}(x^i(t)|x)$ in the online algorithms depending on how likely the scenario's prediction is given the probabilistic supply model and observed supply until t. The online algorithm then returns the next set of jobs to start, s_t, which iteratively defines the full schedule s. We call the two new online algorithms for selecting a set of non-preemptive jobs multi-machine *consensus* and multi-machine *expectation*.

Multi-Machine Consensus.

Our first online algorithm, the multi-machine *consensus* or m-Consensus algorithm, is given in Algorithm 2.

The algorithm solves the offline problem (line 5) for each scenario and then schedules the job that is selected to be started immediately in the likelihood-weighted largest number of scenarios (or none, if more scenarios do not start a new job). This is repeated iteratively, adding one additional job to the schedule at a time, until no more jobs are started. This repetition occurs at most n times (but usually much less frequently), [4] and so including the $\mathcal{O}(n\mathcal{T})$ per call to the offline algorithm, the computational complexity of m-Consensus (for a single time step t) is $\mathcal{O}(n^2\mathcal{N}\mathcal{T})$.

Multi-Machine Expectation.

Multi-machine m-Expectation also relies on sampled scenarios, but uses these to explicitly compute each job's marginal welfare. This promises an economic advantage over m-Consensus, as the expected welfare directly represents the value we wish to maximise (unlike the weighted votes in m-Consensus). However, it also incurs a higher computational cost, as we now evaluate each scenario $|J_r| + 1$ times (once for each available job, and once to evaluate the case where no job is started). Algorithm 3 presents the details of this algorithm, which is largely similar to m-Consensus.

```
1  Algorithm: m-Consensus (J_r, x, s, t)
2  repeat
3      Reset counters f (with −ε for f(⊥))
4      foreach scenario ⟨J^i, x^i⟩ do
5          s' ← Offline(J^i(t) ∪ J_r, x^i(t), s, t)
6          if s'_t = s_t then
7              f(⊥) ← f(⊥) + L(x^i(t)|x)
8          else
9              for j ∈ J_r ∩ s'_t do
10                 f(j) ← f(j) + L(x^i(t)|x)
11     j* ← arg max_{j∈J_r} f(j)
12     if j* ≠ ⊥ then
13         J_r ← J_r \ {j*}; s_t ← s_t ∪ {j*}
14 until j* = ⊥
15 return s
```

Algorithm 2: Schedule the jobs from J_r at t that occur in the most scenarios. Scheduling nothing is denoted by $\bot$.

```
1  Algorithm: m-Expectation (J_r, x, s, t)
2  repeat
3      Reset counters f (with −ε for f(⊥))
4      foreach scenario ⟨J^i, x^i⟩ do
5          f(⊥) ← f(⊥)+
           w(Offline(J^i(t) ∪ J_r, x^i(t), s, t+1)) · L(x^i(t)|x)
6          for j ∈ J_r do
7              s' ← s ; s'_t ← s'_t ∪ {j}
8              f(j) ← f(j)
               +w(Offline(J^i(t) ∪ J_r \ {j}, x^i(t), s', t))
               ·L(x^i(t)|x)
9      j* ← arg max_{j∈J_r} f(j)
10     if j* ≠ ⊥ then
11         J_r ← J_r \ {j*}; s_t ← s_t ∪ {j*}
12 until j* = ⊥
13 return s
```

Algorithm 3: Schedule the jobs from J_r at t that have the highest added value.

4. ONLINE MECHANISM DESIGN

In order to cope with individual agents' incentives we depart from the scheduling paradigm and introduce an incentive-compatible (IC) mechanism. A mechanism can only be IC if allocation is monotonic, in the sense that it should not be possible that an agent[5] reporting a lower type (i.e., a lower valuation, later arrival, earlier departure, or longer job requirement) is allocated instead of an agent reporting a higher type [8].

In order to ensure monotonicity, in this paper we take the approach first proposed in [12], which involves splitting the mechanism at each time step into two phases: precommitment and allocation. Informally, during the precommitment stage, the mechanism checks which agents contribute to a schedule's welfare given the current commitments, and current and future arrivals. If an agent is judged of sufficiently high value, then the mechanism commits to allocating elec-

[3] For now, these are set to $s = \langle\rangle$ and $t = 1$.

[4] In addition, the offline scheduling problem gets smaller by 1 job in every iteration.

[5] The terms 'agent' and 'job' are used synonymously.

tricity to it before its deadline, regardless of the values and demands of future arrivals. Thus, intuitively, an agent has no motivation to misreport its type, e.g., via an earlier deadline or later arrival, because once precommitted, the mechanism guarantees it to be allocated in the future.

During the allocation stage, the actual execution schedule is computed. The focus of this phase is on efficiency only, because incentives issues have already been dealt with during the precommitment phase. Note that, although for computing the allocation we can use the algorithms presented in the preceding section, all resulting schedules must respect the constraints from the precommitment decisions. So, for example, jobs which have been pre committed and are flexible can be delayed, but once their deadline approaches, they must be scheduled, regardless of subsequent arrivals. In the following, we discuss these phases in separate sections.

4.1 Precommitment

In the precommitment stage, the mechanism needs decide which of the jobs to commit to from those that have already arrived, and whether to keep spare capacity for potential future arrivals. In order to ensure monotonicity (and hence IC), jobs that are precommitted must be scheduled before their deadline, regardless of future arrivals, thus precommitments may reduce flexibility of future allocations.

EXAMPLE 2. *Assume the situation described in Example 1 and also assume that either the high-value agent a_3 arrives or supply is not realised. Then, a_1 will not be allocated at t_2 and monotonicity of the allocation would be violated, because if a_1 reported an earlier deadline (i.e., only being available at t_1) he would always be allocated, whereas if he reports his availability throughout both t_1 and t_2 truthfully, then there is a chance of non-allocation. The mechanism will* precommit *to allocate a_1 at t_2, irrespective of the realisation of future supply or future arrivals. In these cases, with small probability, the allocation may be inefficient or the mechanism could make a loss, but, more importantly, monotonicity is guaranteed and incentive compatibility achieved.*

As discussed in [12], in order to guarantee monotonicity, additional constraints need to be imposed on the allocation in the precommitment stage. The most important such constraint (and the only one which is relevant to the discrete-time model used in this paper) is *serialisation*: jobs are first ordered by a monotonicity-respecting criteria, and the precommitment decision is taken by considering jobs sequentially, following this order. Specifically, possible orders that ensure monotonicity in this setting include: decreasing value, increasing length, increasing arrival time (i.e., earlier jobs first), decreasing deadline (i.e., later departures get priority), an increasing rate, as well as combinations of these, such as value density (value divided by length times rate). Tie-breaking rules must also use criteria that guarantee monotonicity.

Essentially, the mechanism considers each job, taken in this order, and considers whether it can fit in a schedule or not, given the previously precommitted jobs. The procedure is formally defined in Algorithm 5: We use decreasing value density, and in case of ties, increasing arrival time. Each unscheduled active job in this order is precommitted if the likelihood-weighted sum of scenario weights in which the job is scheduled, is greater or equal to $\frac{1}{2}$. The scheduling algorithm used is an adaption of the offline scheduler

Algorithm: OFFLINE-PC (P, J, x, s, t)
for $j \in$ SORT(P) `// by decreasing l*r`
do
  $t_{min}, c_{min} \leftarrow$ COSTMINIMALSTARTTIME(j, x, s, t)
  $s_{t_{min}} \leftarrow s_{t_{min}} \cup \{j\}$
$J' \leftarrow \{j \in J \mid j \notin s, d_j \geq t + l_j\}$
for $j \in$ SORT(J')`// by decreasing value density`
do
  $t_{min}, c_{min} \leftarrow$ COSTMINIMALSTARTTIME(j, x, s, t)
  if $c_{min} < v_j$ **then**
    $s_{t_{min}} \leftarrow s_{t_{min}} \cup \{j\}$
return s

Algorithm 4: Heuristically schedule all precommitted jobs (P) and then those unscheduled future jobs (J') that add value, under full knowledge.

Algorithm: m-CONSENSUS-Precommitment(J_r, x, s, t)
for $j \in$ SORT(J_r) `// by decreasing value density`
do
  $f \leftarrow 0$
  foreach *scenario* $\langle J^i(t), x^i(t)\rangle$ **do**
    $s' \leftarrow$ OFFLINE-PC$(P, J^i(t) \cup J_r, x^i(t), s, t)$
    if $j \in s'$ **then**
      $f \leftarrow f + (\mathcal{L}(x^i(t)|x) / \sum_{k\in\{1,..,\mathcal{N}\}} \mathcal{L}(x^k(t)|x))$
  if $f \geq \frac{1}{2}$ **then**
    $P \leftarrow P \cup \{j\}$
return P

Algorithm 5: Decide to commit to jobs that were included in schedules for at least half of the scenarios.

called OFFLINE-PC (Algorithm 4). This algorithm guarantees that all precommitted jobs (denoted by argument P) are scheduled.

THEOREM 1. *The allocation procedure defined in Algorithm 4 and 5 is monotonic, given an assumption of "no early arrivals or late departure"' misreports.*

The proof of monotonicity from [12] applies, with some modifications. Informally, the proof considers each dimension making up a job's type, and shows that an agent's allocation is not worse than under another type which is identical in all dimensions, but is strictly worse in that dimension. So, for example, the allocation of a job with a given value, required amount of electricity and arrival time, but with a later deadline cannot be worse than the allocation of a job with exactly the same parameters, but reporting an earlier deadline.

4.2 Payments

Critical value payments are used to ensure truthfulness of the agents. The critical value of a job in an online mechanism is the minimum value necessary for precommitment given the set of jobs active over the active period of the respective job j [6, p.418]. For a job j with value v_j which is precommitted, its payment $p(j)$ is thus defined as follows.

$$p(j) = \min\{v_{j'} \mid j' \in \text{ schedule } s'\},$$

where s' is the schedule produced by the same algorithm in case j is replaced by j' (with value $v_{j'}$).[6] The payment

[6]Note that truthfulness entails $p(j) \leq v_j$.

Algorithm: m-Consensus-Allocation(P, x, s, t)
repeat
  Reset counters f (with $-\epsilon$ for $f(\perp)$)
  foreach *scenario* $\langle J^i(t), x^i(t)\rangle$ **do**
    $s' \leftarrow$ Offline-PC $(P, J^i(t), x^i(t), s, t)$
    if $s'_t = s_t$ **then**
      $f(\perp) \leftarrow f(\perp) + \mathcal{L}(x^i(t)|x)$
    else
      for $j \in P \cap s'_t$ **do**
        $f(j) \leftarrow f(j) + \mathcal{L}(x^i(t)|x)$
  $j^* \leftarrow \arg\max_{j \in P} f(j)$
  if $j^* \neq \perp$ **then**
    $P \leftarrow P \setminus \{j^*\}$; $s_t \leftarrow s_t \cup \{j^*\}$
until $j^* = \perp$
return s

Algorithm 6: Schedule the precommitted jobs P at t that occur in the most scenarios, under the condition that they all are eventually allocated.

Algorithm: m-Expectation-Allocation(P, x, s, t)
repeat
  Reset counters f (with $-\epsilon$ for $f(\perp)$)
  foreach *scenario* $\langle J^i(t), x^i(t)\rangle$ **do**
    $f(\perp) \leftarrow f(\perp) +$
    $w($Offline-PC$(P, J^i(t), x^i(t), s, t+1)) \cdot \mathcal{L}(x^i(t)x)$
    for $j \in P$ **do**
      $s' \leftarrow s$; $s'_t \leftarrow s'_t \cup \{j\}$
      $f(j) \leftarrow f(j) + w($Offline-PC$(P \setminus \{j\}, J^i(t),$
      $x^i(t), s', t)) \cdot \mathcal{L}(x^i(t)|x)$
  $j^* \leftarrow \arg\max_{j \in P} f(j)$
  if $j^* \neq \perp$ **then**
    $P \leftarrow P \setminus \{j^*\}$; $s_t \leftarrow s_t \cup \{j^*\}$
until $j^* = \perp$
return s

Algorithm 7: Schedule the precommitted jobs P at t that give the highest added value, under the condition that they all are eventually allocated.

thus is not just based on demand and supply at the moment of precommitment, but also at later times (until its latest starting time). This is done, since otherwise an agent could report a later arrival time and reduce its payment. A consequence of this approach is that under specific conditions there is a chance that the received payments are not sufficient to cover the cost of conventional generation (i.e., in case not much renewable supply is available, the prediction was optimistic, and there is not much competition among jobs) incurred by the mechanism. This cannot be remedied without harming the efficiency of the schedule.

4.3 Allocation

As long as all precommitted jobs are scheduled, we are free to use any algorithm in the allocation phase. In this paper, we stay close to the online algorithms we described in Section 3. To force scheduling of all jobs in P in Algorithms 6 and 7, we replace the offline scheduler by Offline-PC, and only select jobs from P to schedule at t.

While the introduction of a precommitment phase was done to make the mechanism IC, it does not necessarily lead to worse results. This is illustrated in the following example.

Example 3. *We consider a situation without flexibility and compare the choices made by the mechanism to those by multi-machine* consensus *(Algorithm 2). Suppose $t = 0$, the following three active jobs have the current time as arrival time, a consumption rate of 1, and the following lengths l_j and values v_j. (The deadline then is exactly equal to the length l_j.)*

jobs

j	v_j	l_j	v_j/l_j
1	9	3	3
2	1	1	1
3	3	2	3/2

scenarios

i	$t=0$	$t=1$	$t=2$
1	2	2	0
2	2	2	0
3	2	2	1
4	2	2	2

Suppose there are four scenarios, which all include the current supply of 2, and in some cases slightly different future supplies. Additionally, each scenario includes a virtual job ($j = 4$) of value 4 and length 2, to be expected at $t = 1$. We assume the cost of conventional generation is 10 per time slot for a production rate of 1.

*The decision for m-*Consensus *is made by repeatedly scheduling all jobs in all scenarios (with a greedy heuristic, based on value density), and then starting the job that occurs in most schedules. In this example, $j = 1$ is scheduled in scenarios 3 and 4, $j = 2$ is scheduled in scenario 1,2, and 4 (the latest because the virtual job can then be included), and $j = 3$ is scheduled in scenario 1,2, and 3. Therefore jobs 2 and 3 are scheduled by m-*Consensus*, for a total value of 4 and an expected value of 6 (there is a 50% chance that virtual job 4 can be executed).*

*The decision by m-*Consensus*-Precommitment is done per job, heuristically ordered by value density. A job is precommitted if at least half of the likelihood-weighted scenarios would schedule it. Job 1 meets this criterion and thus is committed first; job 3 then follows. This schedule has a value of 12, but a 50% chance of a cost of 10, which gives it an expected value of 7, which exceeds the value of the m-*Consensus *schedule. This example can be generalised as long as conventional generation is more expensive than any value density.*

5. EVALUATION

This section presents the empirical evaluation of the algorithms. The goal is to compare the performance of the algorithms and mechanisms in terms of their efficiency with an offline optimal scheduler using perfect information.[7] The corresponding objective function is given in Equation 1. To this end, we vary the number of scenarios used by the algorithms, evaluate the benefit of using a model of future demand and, most importantly, vary job flexibility to see if the algorithms can take advantage of this flexibility. Furthermore, we quantify efficiency losses due to precommitment, which is required to ensure truthfulness. Finally, we verify the computational complexity of the algorithms.

5.1 Experimental Setup

We consider settings with 10 and 30 scenarios and use two variants: in experiments with a model, each scenario includes information on future demand and supply; in experiments with no model, each scenario only includes information about future supply. Thus, we quantify the value of

[7]We use Gurobi 5.5 to compute offline-optimal with a 1% MIP gap.

uncertain information about future demand in terms of efficiency. We use the model below to generate both the scenarios as well as the actual realisation independently.

Demand.

As introduced in Section 2.1, jobs $j \in J$ are characterised by $\langle v_j, r_j, l_j, a_j, d_j \rangle$. For simplicity, the consumption rate is set to $r_j = 1, \forall j \in J$ in the experiments. Job length l is sampled from a uniform distribution over $\{1, 2, \ldots, 6\}$ and valuations v from a uniform distribution over the real interval $[0, 10]$.

Jobs are generated over a 24-hour period, with new jobs being generated each hour. In our experiments, we choose to keep total demand fairly constant, to reduce variation and reflect the fact that demand can typically be predicted with fairly high accuracy (in contrast to renewable supply). To achieve this, at each time step t, we draw a value y uniformly from $\{3, 4, 5, 6, 7\}$. Assuming all jobs start on their arrival, if current demand for electricity, y', is less than y, we generate $y - y'$ additional jobs. Otherwise, we generate no jobs. This ensures that, if jobs are executed immediately, at any point in time, total demand ranges between 3 and 7 units. So far, we have not considered job flexibility, which is defined as $d_j - a_j - l_j$. In our experiments, job flexibility is a control parameter, and so we set this value equal for all jobs. We vary flexibility between 0 (no flexibility) and 5 time steps.

Renewable Supply.

To realistically model uncertain supply of renewable energy, we use publicly available historical wind data from the Sotavento wind farm in Galicia, Spain.[8] This wind farm consists of 24 turbines, with a combined output of up to 17.56MW. However, in order to scale the available supply in our experiments, we model only the wind speed and derive the corresponding power supply using a sigmoid power curve that is based on the installed turbine technology [10]:

$$p_r(w_t) = C \cdot (1 + \mathrm{e}^{6 - \frac{2}{3} w_t})^{-1}, \tag{2}$$

where $p_r(w_t)$ is the available power from wind generation given the wind speed w_t at time t. Here, C is a factor that we use to scale supply and that corresponds to capacity of the installed wind generators. Specifically, in each experiment installed capacity C is scaled such that total supply from wind equals the total amount of energy demanded [13], i.e., $\sum_t p_{r,t} = \sum_{j \in J} q_j$.

We use the wind data in two ways — first, for a single run of our experiment, we select a random subsequence of wind speed data to generate the actual supply available during that run. Second, we train a generative probabilistic model on the data set, in order to allow our scheduling mechanisms to generate new scenarios, and to revise the likelihood of scenarios given new information as it becomes available. We use only the first two years of data to train this model and the remaining data to generate the realisations.

In more detail, we use a hidden Markov model [5] with ten hidden states as our generative model, as this yields good results in practice on the wind data. For training the model, we use the expectation-maximisation algorithm, and employ the forward-backward algorithm for inference.

[8]This data is available from `www.sotaventogalicia.com`, and we use hourly data from May 2008 to 2013.

Conventional Generation.

Conventional generation (CG) as a source of reliable backup generation is necessary in order for the non-preemptive and precommitted jobs to be served even in the case of an unanticipated shortfall of renewable generation. We assume CG to be characterised by constant marginal cost, i.e., $c_c(p_c) = b \cdot p_c$, and we set b to a value approximately 30% above average job value density. With this cost parameter, low-valued jobs should not be served if there is insufficient renewable generation. On the other hand, if there is some, but insufficient renewable generation to fully serve a job from renewable generation, social welfare benefits if the remaining part is served from CG instead of rejecting the job. As job flexibility is increased, the amount of CG used by the algorithms can be expected to decrease.

5.2 Results

Our experimental results are illustrated in Figure 1. We vary job flexibility between zero and five time steps (hours) and social welfare is normalised by offline-optimal assuming job flexibility of five hours. Furthermore, the offline-optimal is given by a solid black line. The reported results are the mean relative social welfare, and the error bars represent the corresponding standard deviations. Different colours indicate the type of algorithm (*consensus* or *expectation*), while linetype and point shape indicate the presence (or absence) of a demand model. We separate our results of 120 repetitions by the number of scenarios (rows) and whether a scheduling or mechanism approach (columns) is followed.

First, social welfare initially increases with flexibility for all algorithms. Furthermore, the mechanisms achieve only slightly reduced welfare compared to the extended algorithms which rely on the strong assumption of cooperative agents.

Second, the availability of a demand model is more beneficial for *expectation* than *consensus*-based algorithms. Regarding the *schedulers*, only m-EXPECTATION improves upon the model-free variant at higher values of flexibility, while m-CONSENSUS seems to perform better without a model. However, the loss from using a model in this case decreases over flexibility. In contrast, both types of *mechanisms* benefit from using a demand model at higher flexibility values. The value of using a model is especially pronounced when m-EXPECTATION is used in the allocation phase. At low flexibility levels, not using a model of future demand can be slightly advantageous for both, schedulers and mechanisms. This seems to be associated with the problem of under-commitment when using a model, as the value of future jobs might be overestimated. At higher flexibility levels, however, this drawback is (mostly) compensated by better allocation decisions based on information from the model.

Third, and most notably, the average gap in social welfare, i.e., the cost of incentive compatibility in our settings does not exceed 3%; while m-EXPECTATION achieves approximately 90% of the offline optimal at five hours flexibility, the corresponding mechanism achieves 87%.

Our results show that computation time increases linearly in job flexibility. Computing the hourly schedules for one day with 30 scenarios not using a demand model and flexibility set to 5 time steps takes about 10 seconds (approximately 80 seconds with model). m-EXPECTATION is computationally more involved (2 minutes without and 7 minutes with a demand model). Interestingly, the mechanism using *expectation* in the allocation phase is computationally less

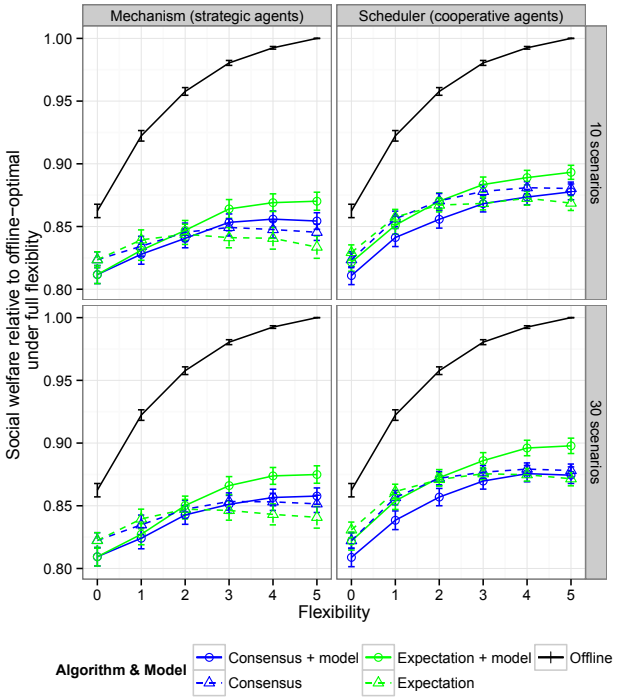

Figure 1: Social welfare relative to offline-optimal (at flexibility set to 5 time steps) over flexibility.

involved for large flexibilities than its scheduling counterpart: the *consensus*-based precommitment step reduces the number of jobs under consideration at each time step.

6. CONCLUSIONS

In this paper, we extend the *Expectation* and *Consensus* algorithms to cope with multi-unit, non-preemptive demand under uncertainty on both supply and demand side. Specifically, to deal with uncertain supply, instead of using equally weighted scenarios, we use the likelihood of each scenario given past observations to adjust the weights of each scenario online. By doing so, we can incorporate new information without re-sampling the scenarios. Furthermore, in order to apply the principles of these algorithms to settings with self-interested agents, we use the concept of precommitment to achieve monotonicity and thus incentive-compatibility for the demand side. Finally, non-preemptive jobs and precommitments can only be scheduled if there are guarantees in terms of the availability of future supply. To deal with this problem, we consider two supply sources: cheap but uncertain renewables, and costly but unlimited conventional energy. This way jobs can be committed, even if supply is uncertain, and the algorithms are designed to take into account the risk of using the costly alternative.

Our empirical evaluation in the domain of electrical power systems shows a number of interesting results. First, as expected, social welfare increases in job flexibility. Second, using a model of future demand is especially valuable in settings with large flexibilities when using *Expectation*-based approaches. While *Consensus*-based approaches do not achieve the results of *Expectation*-based ones, they are more robust to the lack of a demand model and might be preferred under very tight online time constraints. Nevertheless, *Consensus* benefits from using a model when using precommitment; third, the cost of achieving truthfulness (i.e., by requiring precommitment) is very low and only approximately 3%. We note that our evaluation of the mechanism is placed in the context of the electrical power domain. However, it could be applied to other settings where jobs are non-preemptive and there is a source of free (or cheap), expiring resources and costly backup supply.

In future work we intend to explore the trade-off between economic efficiency and budget deficits for the mechanism. Using the currently proposed mechanism, the system can make a loss when the payments it receives are low (e.g., due to lack of competition on the demand side), while incurring more than expected costs on the supply side. By undercommitting, the mechanism could achieve budget balance (in expectation) at the cost of reduced efficiency as less of the free resource might be used. Another interesting area of future work is considering more computationally efficient methods. Specifically, while the current scenarios are based on a day look-ahead, we would like to explore rolling horizon techniques (e.g., of about 2-3 times the typical job length), and their effect on social welfare.

7. REFERENCES

[1] H. Chang, R. Givan, and E. Chong. On-line scheduling via sampling. In *ICAPS*, pages 62–71, 2000.

[2] F. Constantin and D. Parkes. Self-correcting sampling-based dynamic multi-unit auctions. In *ACM EC*, pages 89–98, 2009.

[3] DECC. UK renewable energy roadmap. Technical report, UK Department of Energy and Climate Change, 2011.

[4] P. Hentenryck and R. Bent. *Online stochastic combinatorial optimization*. The MIT Press, 2009.

[5] B. H. Juang and L. R. Rabiner. Hidden markov models for speech recognition. *Technometrics*, 33(3):251–272, 1991.

[6] N. Nisan, T. Roughgarden, E. Tardos, V. V. Vazirani, and D. C. Parkes, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[7] D. Parkes and Q. Duong. An ironing-based approach to adaptive online mechanism design in single-valued domains. In *AAAI*, pages 94–101, 2007.

[8] D. C. Parkes. Online mechanisms. In N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*, pages 411–439, 2007.

[9] M. Pinedo. *Scheduling: theory, algorithms, and systems*. Springer, 2012.

[10] V. Robu, R. Kota, G. Chalkiadakis, A. Rogers, and N. R. Jennings. Cooperative virtual power plant formation using scoring rules. In *AAAI*, 2012.

[11] J. L. Sawin. Renewables 2013, global status report. Technical report, REN 21, 2013.

[12] S. Stein, E. Gerding, V. Robu, and N. R. Jennings. A model-based online mechanism with pre-commitment and its application to electric vehicle charging. In *AAMAS*, pages 669–676, 2012.

[13] A. Subramanian, M. Garcia, A. Dominguez-Garcia, D. Callaway, K. Poolla, and P. Varaiya. Real-time scheduling of deferrable electric loads. In *American Control Conference*, pages 3643–3650, 2012.