A Family of Resource Constraints for Energy Cost Aware Scheduling

Helmut Simonis and Tarik Hadzic*

Cork Constraint Computation Centre
Department of Computer Science, University College Cork, Ireland
{h.simonis,t.hadzic}@4c.ucc.ie

Abstract. We present a family of resource constraints which allow us to model scheduling problems with time variable resource cost. Our main use case is energy cost minimization, by avoiding peak electricity usage periods in the generated schedule. This at the same time helps to balance the overall demand profile, leading to reduced stress on the national electricity grid. The proposed constraints extend existing cumulative, disjunctive and machine choice constraints. We present abstract formulations of the constraints, some efficient models for cost estimation and describe algorithms for domain pruning based on reduced cost reasoning.

1 Motivation

Time variable electricity tariffs have been introduced in many countries to properly price the cost of electricity generation over changing demand at different times of the day. In Ireland, the All Island Electricity Market (`http://allislandmarket.com`) generates a whole-sale market price in half hour time slots, with prices sometimes varying by a factor of ten between peak and non-peak periods. The high cost at peak utilization is linked to the use of inefficient, expensive stand-by generation plant with an increased carbon footprint. Shifting large-scale customer demand away from peak utilization increases stability of the national grid, and can possibly delay or avoid heavy investment in new generator capacity.

Figure 1 shows demand and price data for two one week periods in January and June 2010. We can see that demand varies significantly during the day, but also from day to day in the week and between seasons. Prices follow the demand, with especially high prices at periods close to the national dispatch limit.

In Ireland, wind power plays an increasing role as a renewable energy source. Unfortunately, its integration in the national grid is problematic, due to the relative isolation of the Irish electricity grid, and wind energy's time variable, and only partially known, supply level. Matching electricity use to changes in wind energy supply is quite difficult to imagine for domestic usage, but can be possible for industrial users, given the right pricing incentive. Widespread

* This work was supported by Science Foundation Ireland (Grant Numbers 05/IN/I886 and 05/IN.1/I886 TIDA 09).

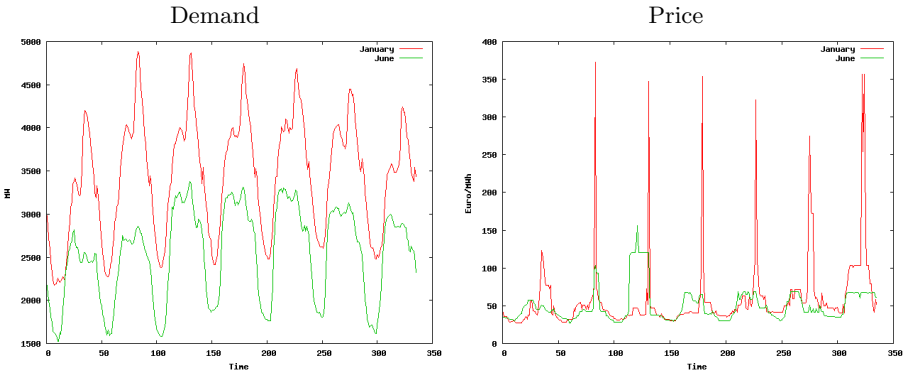


Fig. 1. Electricity Demand and Price, ROI, Comparing January and June data

adaption of electric cars is seen as another way of fully utilizing the wind energy potential, but this is still years in the future.

The current adaption of time variable tariffs for large scale electricity users is still quite limited. This is partially due to the cost of the required smart metering tools and the necessary detailed understanding of electrical usage over short time periods, but also due to a lack of decision support tools which allow the exploitation of time variable tariffs to reduce costs. We are trying to fill this gap using constraint programming.

Our current work is therefore focused on the problem of solving industrial scheduling problems considering time variable energy cost, and using available renewable energy in the most efficient way. This requires agile scheduling tools, which can react quickly to changes in prices, while still considering other optimization criteria like product quality, factory throughput and just-in-time production.

Constraint Programming has been a very successful tool in solving large scale industrial scheduling problems [2,9,10], creating flexible scheduling tools for industries in many domains. Its main advantage over competing technologies is the ease of adapting a system to a changing environment, and the potential to incorporate user-defined strategies and heuristics besides powerful mathematical reasoning techniques.

So far our work [7,8,6] has centered on finding lower bounds on the energy cost for a cumulative schedule. We have defined and compared multiple models and algorithms which can predict the energy cost of a partially defined schedule. The best results use an adaptation of the Linear Programming model for the cumulative constraint from [4]. Experimental results indicate that very good estimates (better than 98% of the optimal value) can be achieved. Obtaining the lower bounds was a required first step in defining constraint filtering rules which remove infeasible values from consideration during search. In this paper we describe these methods for a family of global constraints which combine energy cost

considerations with other scheduling concepts for single or multiple machines, or overall, cumulative resource consumption in a factory. At the same time, we are considering scheduling strategies which can help to find good schedules quickly, without exploring a very large search space completely.

Our current methods are focused on industrial scheduling problems for large scale electricity consumers in manufacturing and other industries. But the underlying technology is also applicable in other areas of significant electricity usage like HVAC (heating/ventilation/air-conditioning) for buildings, or cooling systems in the food industry or for data centres. These are domains where there also is significant overlap with other projects at our institute (4C), and existing collaboration within UCC and with industrial partners.

2 Constraints Family

We consider four variants of the energy cost aware scheduling constraints:

CumulativeCost The total energy consumption is limited by a hard limit, and each task consumes a fixed amount of energy during its execution. The cost is the total cost of energy consumed over time, priced at different levels.

DisjunctiveCost This constraint models a disjunctive machine, where tasks consume different amounts of energy. The order of the tasks on the machine will therefore affect total energy cost.

ParallelMachineCost This considers multiple disjunctive machines, which all contribute to an overall energy limit. Tasks are fixed on their machine, only the order of the tasks can be affected.

MachineChoiceCost We consider multiple disjunctive machines with an overall energy use limit. Tasks can move between machines, with possibly different duration and resource use on every machine.

We now discuss each of the possible constraints in more detail.

2.1 CumulativeCost

We start with the CumulativeCost constraint, which looks at the overall energy consumption of a set of tasks over time. It is an extension of the classical cumulative constraint [1]

$$\texttt{Cumulative}([s_1, s_2, ...s_n], [d_1, d_2, ...d_n], [r_1, r_2, ...r_n], l, p),$$

describing n tasks with start s_i, fixed duration d_i and resource use r_i, with an overall resource limit l and a scheduling period end p. Our new constraint is denoted as

$$\texttt{CumulativeCost}(\texttt{Areas}, \texttt{Tasks}, l, p, \texttt{cost}).$$

The *Areas* argument is a collection of q areas $\{A_1, \ldots, A_q\}$, which do not overlap and partition the entire available resource area $[0, p] \times [0, l]$. Each area A_j has a fixed position x_j, y_j, fixed width w_j and height h_j, and fixed per-unit cost

c_j. Consider the example in Fig. 2 (left). It has 5 areas, each of width 1 and height 3. Our definition allows that an area A_j could start above the bottom level ($y_j > 0$). This reflects the fact that the unit-cost does not only depend on the time of resource consumption but also on its volume. In our electricity example, in some environments, a limited amount of renewable energy may be available at low marginal cost, generated by wind-power or reclaimed process heat. Depending on the tariff, the electricity price may also be linked to the current consumption, enforcing higher values if an agreed limit is exceeded.

We choose the numbering of the areas so that they are ordered by non-decreasing cost ($i \leq j \Rightarrow c_i \leq c_j$); in our example costs are $0, 1, 2, 3, 4$. There could be more than one area defined over the same time slot t (possibly spanning over other time-slots as well). If that is the case, we require that the area "above" has a higher cost. The electricity consumed over a certain volume threshold might cost more.

The *Tasks* argument is a collection of n tasks. Each task T_i is described by its start s_i (between its earliest start $\underline{s_i}$ and latest start $\overline{s_i}$), and fixed duration d_i and resource use r_i. In our example, we have three tasks with durations $1, 2, 1$ and resource use $2, 2, 3$. The initial start times are $s_1 \in [2, 5]$, $s_2 \in [1, 5]$, $s_3 \in [0, 5]$. For a given task allocation, variable a_j states how many resource units of area A_j are used. For the optimal solution in our example we have $a_1 = 2, a_2 = 3, a_3 = 2, a_4 = 0, a_5 = 2$. Finally, we can define our constraint:

Definition 1. *Constraint* `CumulativeCost` *expresses the following relationships:*

$$\forall\ 0 \leq t < p: \quad pr_t := \sum_{\{i | s_i \leq t < s_i + d_i\}} r_i \leq l \tag{1}$$

$$\forall\ 1 \leq i \leq n: \quad 0 \leq \underline{s_i} \leq s_i < s_i + d_i \leq \overline{s_i} + d_i \leq p \tag{2}$$

$$\texttt{ov}(t, pr_t, A_j) := \begin{cases} \max(0, \min(y_j + h_j, pr_t) - y_j) & x_j \leq t < x_j + w_j \\ 0 & otherwise \end{cases} \tag{3}$$

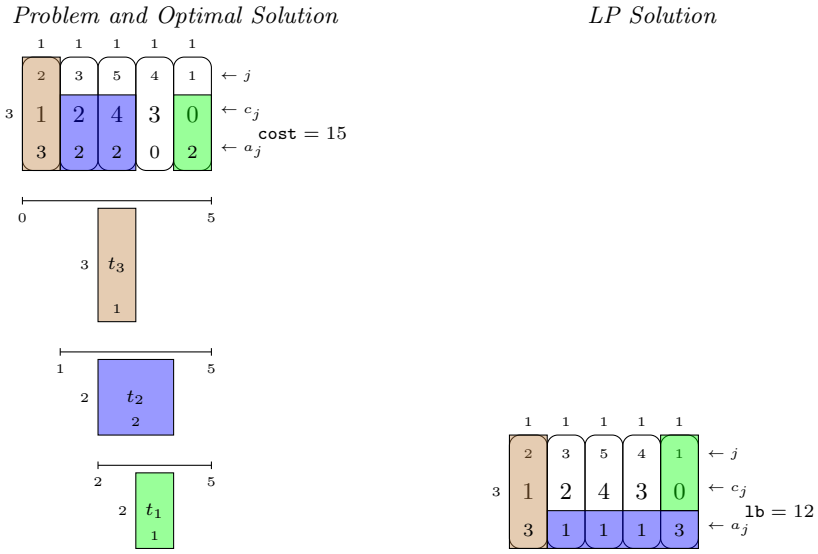
$$\forall\ 1 \leq j \leq q: \quad a_j = \sum_{0 \leq t < p} \texttt{ov}(t, pr_t, A_j) \tag{4}$$

$$\texttt{cost} = \sum_{j=1}^{q} a_j c_j \tag{5}$$

For each time point t we first define the resource profile pr_t (the amount of resource consumed at time t). That profile must be below the overall resource limit l, as in the standard cumulative. The term $\texttt{ov}(t, pr_t, A_j)$ denotes the intersection of the profile at time t with area A_j, and the sum of all such intersections is the total resource usage a_j. The cost is computed by weighting each intersection a_j with the per-unit cost c_j of the area.

Note that our constraint is a strict generalization of the standard cumulative. Since enforcing generalized arc consistency (GAC) for `Cumulative` is NP-hard [4], the complexity of enforcing GAC over `CumulativeCost` is NP-hard as well.

Fig. 2. An example with 3 tasks and 5 areas. Areas are drawn as rounded rectangles at the top, the tasks to be placed below, each with a line indicating its earliest start and latest end. The optimal placement of tasks has cost 15 (left). The LP model produces a lower bound 12 (right).



In [8] we considered different algorithms to compute a lower bound on the resource cost of the CumulativeCost constraint. The best model was based on [4], which describes an LP relaxation of the classical cumulative constraint. We extend this model to handle the cost component directly (equations (6)-(15)).

We introduce binary variables y_{it} which state whether task i starts at time t. For each task, exactly one of these variables will be one (constraint (12)). Equations (11) connect the s_i and y_{it} variables. Continuous variables pr_t describe the resource profile at each time point t, all values must be below the resource limit l. The profile is used in two ways: In (13), the profile is built by cumulating all active tasks at each time-point. In (14), the profile overlaps all areas active at a time-point, where the contribution of area j at time-point t is called z_{jt} (a continuous variable ranging between zero and h_j). Adding all contributions of an area leads to the resource use a_j for area j. This model combines the start-time based model of the cumulative with a standard LP formulation of the convex, piece-wise linear cost of the resource profile at each time point. Note that this model relies on the objective function to fill up cheaper areas to capacity before using more expensive ones. Enforcing the integrality in (8) leads to a mixed integer programming model *DMIP*, relaxing the integrality constraint leads to the LP model *DLP*. The MIP model solves the cumulative-cost constraint to optimality, thus providing an exact bound for the constraint. We can ignore the

actual solution if we want to use the constraint in a larger constraint problem.

$$\texttt{lb} = \min \sum_{j=1}^{q} a_j c_j \tag{6}$$

$$\forall\ 0 \le t < p: \quad pr_t \in [0, l] \tag{7}$$

$$\forall\ 1 \le i \le n, 0 \le t < p: \quad y_{it} \in \{0, 1\} \tag{8}$$

$$\forall\ 1 \le j \le q, \forall\ x_j \le t < x_j + w_j: \quad z_{jt} \in [0, h_j] \tag{9}$$

$$\forall\ 1 \le j \le q: \quad 0 \le \underline{a_j} \le a_j \le \overline{a_j} \le w_j h_j \tag{10}$$

$$\forall\ 1 \le i \le n: \quad s_i = \sum_{t=0}^{p-1} t y_{it} \tag{11}$$

$$\forall\ 1 \le i \le n: \quad \sum_{t=0}^{p-1} y_{it} = 1 \tag{12}$$

$$\forall\ 0 \le t < p: \quad pr_t = \sum_{1 \le i \le n} \sum_{t' \le t < t' + d_i} y_{it'} r_i \tag{13}$$

$$\forall\ 0 \le t < p: \quad pr_t = \sum_{j=1}^{q} z_{jt} \tag{14}$$

$$\forall\ 1 \le j \le q: \quad a_j = \sum_{t=x_j}^{x_j + w_j - 1} z_{jt} \tag{15}$$

2.2 DisjunctiveCost

The `DisjunctiveCost` constraint is the analog generalization of the disjunctive constraint, which allows one task run be run on a machine at any time. Given the areas and tasks as defined before, we can describe the constraint by adding

$$\forall\ i, j | i \ne j: \quad s_i + d_i \le s_j \vee s_j + d_j \le s_i \tag{16}$$

to constraints (1) -(5).

In the LP/MIP model, we extend constraints (6) - (15) with the condition

$$\forall\ 0 \le t < p: \quad \sum_{1 \le i \le n} \sum_{t' \le t < t' + d_i} y_{it'} \le 1 \tag{17}$$

which states that at each time point only one task can be active. Note that the overall resource limit l becomes meaningless, as one only task consumes resources at any time point. The overall resource limit can be checked a priori by comparing it against the resource requirements r_i of the tasks. There is still a minimization problem arranging the tasks in a sequence such that total energy cost is minimal.

This constraint is useful to model a factory which contains a single, disjunctive resource as the main energy consumer. Modelling that machine as a CumulativeCost resource (together with a finite domain disjunctive constraint, say) will lead to a massive underestimation of the energy cost required.

2.3 ParallelMachineCost

In the previous section we considered a situation where a single disjunctive machine dominates the energy consumption. We now consider a case where b disjunctive machines run in parallel, and each task is fixed to one of those disjunctive machines. Let $1 \leq m_i \leq b$ be the machine on which task i is assigned. The `ParallelMachineCost` constraint then consists of constraints (1) -(5) plus the constraints

$$\forall\ 1 \leq k \leq b,\ \forall\ i,j|i \neq j: \quad m_i \neq m_j \vee s_i + d_i \leq s_j \vee s_j + d_J \leq s_i \tag{18}$$

We can express this condition in the LP/MIP model by adding constraints of the form

$$\forall\ 1 \leq k \leq d,\ \forall\ 0 \leq t < p: \quad \sum_{\{i|m_i=k\}} \sum_{t' \leq t < t'+d_i} y_{it'} \leq 1 \tag{19}$$

to constraints (6) - (15).

This `ParallelMachineCost` describes how all tasks compete for energy, and tasks on a single machine must be scheduled with overlap. This model produces stronger bounds than a `CumulativeCost` constraint on the tasks alone, as it considers the disjunctive behaviour as well.

2.4 MachineChoiceCost

The final variant of the constraint family we consider looks at a situation where we have b machines, and tasks can be assigned on alternative machines, possibly with a different duration and resource consumption on each machine. Let d_{ik} be the duration of task i on machine k, and r_{ik} the energy demand for task i on machine k. The variable $m_i \in M_i$ denotes the machine on which task i has been assigned. It must take a value in the set of possible machines M_i for task i.

Definition 2. *Constraint* `MachineChoiceCost` *expresses the following relationships:*

$$\forall\ 0 \leq t < p: \quad pr_t := \sum_{\{i|s_i \leq t < s_i + d_{im_i}\}} r_{im_i} \leq l \tag{20}$$

$$\forall\ 1 \leq i \leq n: \quad 0 \leq \underline{s_i} \leq s_i < s_i + d_{im_i} \leq \overline{s_i} + d_{im_i} \leq p \tag{21}$$

$$\mathtt{ov}(t, pr_t, A_j) := \begin{cases} \max(0, \min(y_j + h_j, pr_t) - y_j) & x_j \leq t < x_j + w_j \\ 0 & otherwise \end{cases} \tag{22}$$

$$\forall\ 1 \leq j \leq q: \quad a_j = \sum_{0 \leq t < p} \mathtt{ov}(t, pr_t, A_j) \tag{23}$$

$$\mathtt{cost} = \sum_{j=1}^{q} a_j c_j \tag{24}$$

We use 0/1 variables y_{itk} to state that task i starts at time t on machine k. If a machine can not run on some machine k, then all entries y_{itk} for that machine will be zero.

$$\texttt{lb} = \min \sum_{j=1}^{q} a_j c_j \tag{25}$$

$$\forall\ 0 \le t < p: \quad pr_t \in [0, l] \tag{26}$$

$$\forall\ 1 \le k \le b, \forall\ 1 \le i \le n, 0 \le t < p: \quad y_{itk} \in \{0, 1\} \tag{27}$$

$$\forall\ 1 \le j \le q, \forall\ x_j \le t < x_j + w_j: \quad z_{jt} \in [0, h_j] \tag{28}$$

$$\forall\ 1 \le j \le q: \quad 0 \le \underline{a_j} \le a_j \le \overline{a_j} \le w_j h_j \tag{29}$$

$$\forall\ 1 \le i \le n: \quad s_i = \sum_{1 \le k \le d} \sum_{t=0}^{p-1} t y_{itk} \tag{30}$$

$$\forall\ 1 \le i \le n: \quad \sum_{1 \le k \le b} \sum_{t=0}^{p-1} y_{itk} = 1 \tag{31}$$

$$\forall\ 0 \le t < p: \quad pr_t = \sum_{1 \le k \le b} \sum_{t' \le t < t' + d_{ik}} y_{it'k} r_{ik} \tag{32}$$

$$\forall\ 0 \le t < p: \quad pr_t = \sum_{j=1}^{q} z_{jt} \tag{33}$$

$$\forall\ 1 \le j \le q: \quad a_j = \sum_{t=x_j}^{x_j + w_j - 1} z_{jt} \tag{34}$$

$$\forall\ 1 \le k \le b,\ \forall\ 0 \le t < p: \quad \sum_{1 \le i \le n} \sum_{t' \le t < t' + d_{ik}} y_{it'k} \le 1 \tag{35}$$

3 Pruning and Search

The descriptions of the constraints above have provided a method to compute a lower bound on the cost for each constraint with an LP model. We can use the solution of the LP model to remove inconsistent values in the domains via reduced cost filtering [3,5]. Given a current upper bound on the total cost $\overline{z}$, the current LP solution z^* and reduced costs $\overline{c_{it}}$ for a variable y_{it} in the LP solution, we observe that

$$z^* + \overline{c_{it}} > \overline{z} \implies y_{it} = 0 \tag{36}$$

This means that we can remove start time t from the domain of task i, since every solution which would use that start time would have a total cost higher than the current upper bound.

The reduced costs can also provide the basis for a search strategy in the finite domain solver. We should try to assign those start times which correspond to small reduced costs first. Note that this requires that we make the costs available outside the constraint itself, which we haven't implemented at this point.

4 Experiments

In this section we present a series of experiments with cumulative, disjunctive and parallel disjunctive machine instances, where a set of n tasks is distributed over M machines. We use 48 half-hour time periods with area costs drawn randomly from $[0, 100]$. We use a time resolution of 10 minutes, this means that each area is 3 units wide, leading in total to a horizon of $p = 144$.

4.1 Comparing `CumulativeCost` and `ParallelMachineCost`

In table 1, we show results for the `CumulativeCost` model. The columns indicate: n, the number of tasks, u_{cum}, the cumulative utilization percentage, the period length p, and the resource limit l, set to achieve a desired utilization factor. The column *sat* states how many of the generated problems were satisfiable, C_{mip} is the optimal cost. The factor q_{dlp} is the ratio of LP to MIP solution, t_{mip} the time (in ms) to solve the MIP problem, and t_{dlp} the time to set up the finite domain and LP model. We show the minimum (pr^{min}), average (pr^{avg}), and maximal (pr^{max}) percentage of domain values removed by the reduced cost filtering.

Table 1. `CumulativeCost` Reduced Cost Filtering. Random area costs. 10-minute resolution. $d_{max} = 6$.

n	u_{cum}	p	l	sat	C_{mip}	q_{dlp}	t_{mip}	t_{dlp}	pr^{min}	pr^{avg}	pr^{max}
40	20	144	22.0	100.0	7022	99.93	24596	122	86.24	93.862	96.43
40	30	144	13.9	100.0	9198	99.86	145835	95	78.35	86.818	94.21
40	40	144	10.5	100.0	12303	99.447	2008137	100	33.157	66.77	89.11
80	20	144	42.5	100.0	13469	99.99	2915	201	85.33	94.01	96.40
80	30	144	30.3	100.0	17377	99.99	3391	174	88.58	92.87	94.28
80	40	144	21.1	100.0	25818	99.99	24447	188	83.18	87.14	90.13
80	50	144	17.7	100.0	28289	99.99	302892	207	62.24	76.62	89.23
120	20	144	66.7	100.0	24169	100.0	2068	303	93.74	95.11	96.55
120	30	144	42.6	100.0	30049	99.99	12426	278	85.59	91.48	93.53
120	40	144	32.4	100.0	37521	99.99	287266	280	72.17	84.21	91.80
120	50	144	26.3	100.0	46222	99.99	1341691	304	58.03	69.95	86.47
160	20	144	85.7	100.0	30111	99.99	2983	400	93.33	94.75	95.46
160	30	144	54.5	100.0	37407	99.99	14541	380	86.28	90.40	93.36
160	40	144	42.7	100.0	50704	99.99	58071	402	75.87	86.13	90.71
160	50	144	34.2	100.0	63507	99.99	1347527	454	60.85	71.66	83.34

All results are averages over ten runs. We see that the MIP times increase quite quickly with the resource utilization, while the LP solving times stay within a few hundred milliseconds. At the same time, the quality of the LP solution is exceptional: In all cases, its cost is very close to the optimal solution. The amount of pruning that can be achieved by reduced cost filtering is quite significant. If we use the MIP solution as upper bound, we can remove on average between

66% and 95% of all domain values at the root node. This is of course unrealistic, as this assumes a priori knowledge of the optimal solution. But we could start for example with a slight relaxation of the LP bound to obtain a very high quality initial solution.

Table 2 shows the corresponding results for parallel disjunctive machines, each running 40 tasks. The results with 80 tasks therefore correspond to 2 machines in parallel, 120 tasks to 3 machines, and 160 tasks to 4 machines in parallel, as indicated in column M. Columns u_{mul}^{avg} and u_{mul}^{max} give the average and maximal utilization of the disjunctive machines, considering only the sum of durations of the tasks allocated to each machine. Given the random choice of task durations, it is possible to create instances with more than 100% utilization on one of the parallel machines. These instances are infeasible, we disregard them and run until we have found 10 feasible instances. The *sat* column gives an indication of the percentage of satisfiable instances found for each set of parameters.

Table 2. `ParallelMachineCost` Reduced Cost Filtering. Random area costs. 10-minute resolution. $d_{max} = 6$.

n	M	u_{cum}	u_{mul}^{avg}	u_{mul}^{max}	p	l	sat	C_{mip}	q_{dlp}	t_{mip}	t_{dlp}	pr^{min}	pr^{avg}	pr^{max}
80	2	20	90.87	93.82	144	40.0	28.57	33765	99.95	2087	412	73.06	84.639	93.16
80	2	30	93.68	95.56	144	27.4	21.74	41967	99.96	2713	379	73.26	84.168	89.54
80	2	40	93.33	96.46	144	20.7	20.00	40538	99.95	2156	401	77.92	82.104	86.94
80	2	50	93.02	95.21	144	16.3	19.61	38924	99.91	3117	463	42.78	77.929	91.70
120	3	20	92.89	96.81	144	62.0	52.63	60156	99.94	5078	979	63.46	76.38	87.43
120	3	30	92.29	97.22	144	40.2	6.58	56410	99.95	5185	927	68.19	78.19	92.53
120	3	40	91.11	94.93	144	30.0	1.81	56779	99.95	5185	918	65.89	78.54	91.44
120	3	50	91.64	96.11	144	24.3	2.21	56311	99.94	6894	1095	68.32	76.66	87.89
160	4	20	92.55	97.29	144	80.9	2.96	71846	99.96	8286	1464	68.43	77.52	88.17
160	4	30	93.99	98.19	144	56.4	2.79	82362	99.96	7672	1460	62.23	76.02	89.75
160	4	40	93.94	97.71	144	42.0	2.11	78011	99.95	10944	1341	60.78	73.18	85.09
160	4	50	94.18	98.68	144	33.1	7.87	80445	99.94	11038	1490	54.44	69.14	82.96

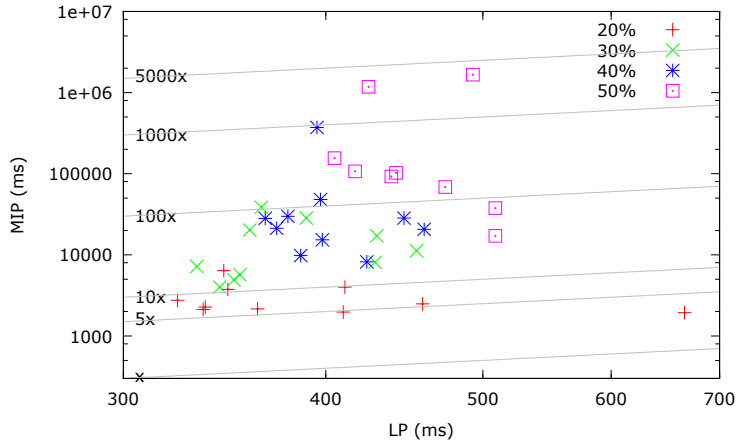
The run-time results in table 2 are somewhat different compared to the cumulative case. The MIP times increase more slowly with utilization and number of tasks, while the LP times are significantly longer. This means that the ratio between MIP and LP is much smaller than in the cumulative case. The solution quality, on the other hand, is very similar: The initial LP solution is very close to the optimal MIP cost, and the reduced cost pruning again can eliminate a large percentage of the domain values in the root node.

Note the difference in the optimal cost value C_{mip} in the cumulative and the parallel machine case for matching scenarios. Using the cost of the cumulative model as an approximation of the parallel machine case would lead to a very large under-estimation of the cost, i.e. it would be far too optimistic. This shows

why we need this complete family of constraints, instead of just combining a `CumulativeCost` constraint with several standard `Disjunctive` constraints.

Figures 3 and 4 show more detailed results for the `CumulativeCost` and `ParallelMachineCost` results for 160 tasks (four machines in parallel for the `ParallelMachineCost` model). We compare the run time of the DLP and the MIP models at the root node for different cumulative utilization rates from 20% to 50%. For the `CumulativeCost` model we can see a clear increase in time for both LP and MIP with increased utilization, with MIP times up to 5000 times longer than LP solving times. For the `ParallelMachineCost` model the ratio MIP/LP is much smaller, and there is no easily seen link between utilization and execution times. This seems to indicate that the disjunctive restrictions seem to dominate for these utilization values.

Fig. 3. LP/MIP ratio for cumulative problem: 160 tasks, time resolution 10 min, cumulative utilisation between 20% and 50%



4.2 Real-world Cost Data

Tables 3 and 4 repeat some of the experiments using real-world cost data for electricity prices from Ireland. Overall, results are quite similar, but there is one entry (80 tasks, 40% utilisation) in table 4 which needs further investigation.

4.3 Comparing `CumulativeCost` and `DisjunctiveCost`

We also ran some experiments to compare `CumulativeCost` and `DisjunctiveCost` models with $n = 40$ tasks. For the disjunctive case the cumulative resource limit has no effect, as long as it is chosen to be larger than r_{max}, since only a single

Fig. 4. LP/MIP ratio for multiple disjunctive problem: 4 machines, 160 tasks, time resolution 10min, cumulative utilisation between 20% and 50%

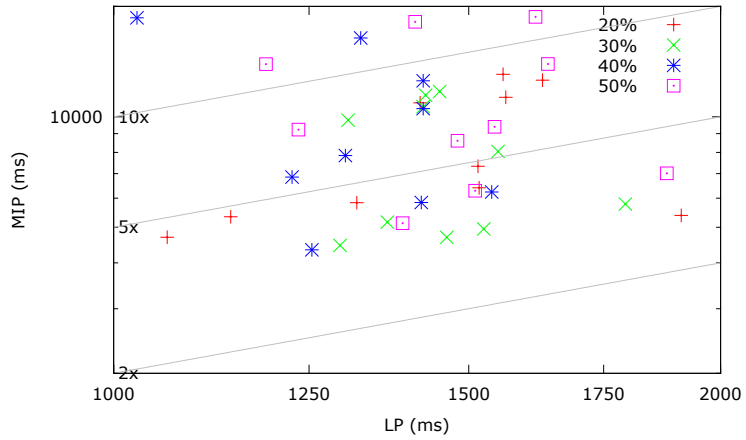


Table 3. `CumulativeCost` Reduced Cost Filtering. Real-world area costs. 10-minute resolution. $d_{max} = 6$.

n	u_{cum}	p	l	sat	C_{mip}	q_{dlp}	t_{mip}	t_{dlp}	pr^{min}	pr^{avg}	pr^{max}
80	20	144	44.8	100.0	16207	99.996	3991	197	88.60	94.41	97.21
80	30	144	29.7	100.0	16301	99.999	5210	175	81.85	90.48	95.19
80	40	144	21.3	100.0	24078	99.991	13967	181	74.82	85.30	90.01
160	20	144	86.5	100.0	34019	99.999	3909	397	92.58	94.84	96.51

Table 4. `ParallelMachineCost` Reduced Cost Filtering. Real-world area costs. 10-minute resolution. $d_{max} = 6$.

n	M	$_{cum}$	u^{avg}_{mul}	u^{max}_{mul}	p	l	sat	C_{mip}	q_{dlp}	t_{mip}	t_{dlp}	pr^{min}	pr^{avg}	pr^{max}
80	2	20	93.99	96.60	144	40.5	15.39	32384	99.99	2736	381	76.04	79.35	83.84
80	2	30	93.61	95.83	144	28.2	8.20	35306	99.99	2896	328	70.89	78.50	83.07
80	2	40	92.57	95.07	144	20.1	6.94	32620	89.99	2322	320	0.0	72.64	83.84
80	2	50	93.12	94.93	144	16.2	9.90	33995	99.98	3117	341	69.119	76.30	81.29
160	4	20	92.26	97.57	144	81.9	5.85	65034	99.987	8326	1223	70.06	75.14	79.50
160	4	30	93.21	98.26	144	56.0	1.79	68825	99.98	11192	1267	64.06	73.73	78.04
160	4	40	93.28	97.85	144	41.6	2.94	66701	99.99	11983	1264	70.55	75.88	81.54
160	4	50	92.69	98.19	144	33.3	2.10	65999	99.98	13850	1345	68.98	74.37	78.84

task can be scheduled at any one time, and no overlap is possible. We therefore set the resource limit to $l = r_{max} + 1$ in both cases. In both cases we use random area costs and 10-minute resolution. The maximal duration of a task is $d_{max} = 6$.

Table 5. Comparing `CumulativeCost` with `DisjunctiveCost` for 40 tasks

type	n	u_{cum}	u_{mul}	p	l	sat	C_{mip}	t_{mip}	t_{dlp}	q_{dlp}	pr^{min}	pr^{avg}	pr^{max}
disj	40	-	91.6	144	9	66.667	17731	896	156	99.95	81.272	87.821	93.618
cumul	40	50.8	-	144	9	100.0	14478	138736.6	147.6	98.26	5.50	41.25	79.08

Even given the limited sample size, we see that the cumulative problem is much harder to solve for MIP than the disjunctive problem, while the LP runtimes are comparable. Also note that even with this very restrictive resource limit, the cumulative cost is still significantly lower. This is another indicator that we really need all of the proposed variants to obtain good cost estimates, and thus good domain pruning.

5 Summary and Future Work

We have presented a family of resource constraints which allow us to model scheduling problems with time variable resource cost. Our main use case is energy cost minimization, by avoiding peak electricity usage periods in the generated schedule. The proposed constraints extend existing cumulative, disjunctive and machine choice constraints. Initial experiments show that reduced cost filtering can significantly restrict domains of the start variables of our constraints, while the LP relaxation provides very good cost estimates.

We did not have time to experiment with the `MachineChoiceCost` model yet. We expect this to be significantly harder to solve as the number of machines increase, since the number of variables y_{itk} per task increases with b, the number of machines.

We also want to study the incremental solution of the LP model as the finite domain solver branches through its search space. The times given here are for the root node only, but we do not expect to resolve the complete problem from scratch at every search node, especially as the reduced cost filtering already removes a large percentage of the domain values in the first step. This should mean that the times given for the LP model do not have to be multiplied with the number of choices taken in order to estimate the time needed to find the complete solution.

References

1. A. Aggoun and N. Beldiceanu. Extending CHIP in order to solve complex scheduling problems. *Journal of Mathematical and Computer Modelling*, 17(7):57–73, 1993.
2. P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*. Kluwer, Dortrecht, 2001.
3. Filippo Focacci, Andrea Lodi, and Michela Milano. Cost-based domain filtering. In Joxan Jaffar, editor, *CP*, volume 1713 of *Lecture Notes in Computer Science*, pages 189–203. Springer, 1999.
4. John Hooker. *Integrated Methods for Optimization*. Springer, New York, 2007.
5. Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., New York, NY, USA, 2006.
6. H. Simonis and T. Hadzic. Constraint-based scheduling for reducing peak electricity use. In *CompSust'10: 2nd International Conference on Computational Sustainability*, Boston, MA, June 2010. `http://4c.ucc.ie/~hsimonis/sustain.pdf`.
7. H. Simonis and T. Hadzic. An energy cost aware cumulative. In *Third International Workshop on Bin Packing and Placement Constraints BPPC'10*, Bologna, Italy, June 2010. `http://4c.ucc.ie/~hsimonis/tidacpaiorabstract.pdf`.
8. H. Simonis and T. Hadzic. A resource cost aware cumulative. In *The 9th International Workshop on Constraint Modelling and Reformulation (ModRef 2010)*, September 2010. (to appear) `http://4c.ucc.ie/~hsimonis/tida1.pdf`.
9. Helmut Simonis. Building industrial applications with constraint programming. In Hubert Comon, Claude Marché, and Ralf Treinen, editors, *CCL*, volume 2002 of *Lecture Notes in Computer Science*, pages 271–309. Springer, 1999.
10. Helmut Simonis. Models for global constraint applications. *Constraints*, 12(1):63–92, 2007.