The Ordered Disjunctive-Cost Constraint

No Author Given

No Institute Given

Abstract. Scheduling problems involving energy-consuming tasks are of increasing importance. The Minimum Cost JobShop Problem is introduced as a generalization of the standard JobShop to include energy costs that depend both on the start time and duration of activities. This problem is modeled and solved with a disjunctive global constraint for computing the energy cost when the order of tasks is fixed. A domain consistent filtering algorithm for this constraint running in $\mathcal{O}(n \cdot H)$ and based on a dynamic program is proposed. We compare the performances of the global constraint with a formulation based on the Resource-Cost AllDifferent global constraint. Results illustrate our new filtering drastically reduces the number of search nodes, leading to an important speedup.

Keywords: Scheduling, Dynamic Programming, Global Constraint, Task Sequence, Minimum Cost JobShop, Energy Costs, Constraint Programming

1 Introduction

The Constraint Programming (CP) technology is well-known to be successful for Constraint-based scheduling [1, 9]. Optimization traditionally focuses on *time* characteristics of the schedule (e.g., Makespan, Total Tardiness, Maximum Lateness). In recent years, the *cost* of a schedule became another important facet for scheduling problems where energy-consuming tasks are involved [13, 3, 17]. This happens in the context of a more prominent usage of renewable energies [21], leading to a larger fluctuation of energy prices. Scheduling most energy-consuming tasks when the prices are low provides a competitive advantage. The production planning should therefore leverage as much as possible the flexibility in the production process to translate it into energy cost savings.

Some problems involve the execution of a set of tasks in a given sequence. For instance, the tasks within a job in a JobShop or a FlowShop must be executed in a fixed order. This also occurs in continuous casting problems [8, 17].

This paper concentrates on scheduling problems where 1) sets of tasks are constrained to be ordered in a given sequence; 2) tasks have a fixed resource cost if started at a given time slot ; and 3) the objective is to find a schedule of minimum total resource cost. In particular, we introduce the Minimum Cost JobShop Problem (MCJP), a generalization of the JobShop problem with energy-consuming tasks.

Existing approaches compute lower bounds on the total cost of a set of tasks by assuming tasks are preemptive (i.e., can be interrupted after having started)

[17]. This is a weak relaxation, especially when the tasks are known to be ordered. This paper solves this problem with a dynamic program that computes the exact minimum cost of a set of in-sequence non-preemptive tasks. A by-product of the dynamic program is a table containing optimal values to subproblems. This table permits to compute exactly the induced minimum costs for all the variable-value assignments, instrumental for implementing domain-consistent or bound-consistent propagators. We compare the filtering obtained with the state-of-the art on the MCJP. Results illustrate our new filtering drastically reduces the number of search nodes, leading to an important speedup. From our experiments, the domain-consistent filtering only slightly decreases the number of nodes as compared with the bound-consistent propagator.

Related Work Global constraints considering (electricity) costs have been defined for different kinds of resources [13]. Several models to compute lower bounds on the cost for the CumulativeCost are compared in [14]. The TaskIntersection [20] global constraint was introduced to restrict the total intersection of (variable duration) tasks with a set of fixed non-overlapping intervals $\mathcal{I}$. The cost of starting a task at a given time point therefore depends on the length of the task and on the set $\mathcal{I}$. The ResourceCostAllDifferent (RCAD) constraint [17] possesses an efficient filtering algorithm for problems where the cost of a variable is the product of its consumption with the price of the value it is assigned to. This is a particular case of the GCCCost global constraint [11, 12, 5]. RCAD has been shown to be efficient for scheduling problems with energy-consuming tasks. Yet, the filtering assumes the tasks are preemptive, weakening the computed lower bounds and the filtering. Finally, a recent application of CP was successfully used to optimize a tissue manufacturing planning problem from an energy viewpoint [3].

Background Non-preemptive scheduling problems are usually modeled in CP by associating three variables to each task i: s_i, e_i, and p_i representing respectively the starting time, completion time, and processing time of i. These variables are linked together by the following relation: $s_i + p_i = e_i$. Depending on the problem, the scheduling of the tasks can be restricted by the availability of different kinds of resources required by the tasks. Each task i is constrained by a release date r_i and a deadline d_i, i.e., $s_i \in [r_i, d_i - p_i]$ and $e_i \in [r_i + p_i, d_i]$. We denote the domain of a variable X as $D(X)$, and the minimum and maximum of the domain of a variable X are written $\underline{X}$ and $\overline{X}$, respectively. We define $c_{i,t}$ as the cost for starting a given task i at the time slot t ($c_{i,t} = +\infty$ if $t \notin D(s_i)$).

Paper Outline We first introduce the Minimum Cost JobShop Problem (MCJP), a motivating problem for our new global constraint, OrderedDisjunctiveCost, that is formally defined in the subsequent section. We then describe a dynamic program to compute a sharp lower bound on the total cost of a sequence of consuming tasks. The output of this dynamic program allows computing exactly the induced variable-value assignment cost in $\mathcal{O}(1)$, leading to straightforward infer-

ence rules. Finally, our experimental results show our approach can drastically reduce the number of search nodes, so that a significant speedup is obtained.

2 The Minimum Cost JobShop Problem

The Minimum Cost JobShop Problem (MCJP) is a simple extension of the JobShop Problem with energy-consuming tasks. It serves as a motivation for our work, yet the constraint we introduce can be used in other problems.

Problem Definition A set $\mathcal{J}$ of jobs, each made of N tasks, must be scheduled on a set $\mathcal{R}$ of resources. At each time slot, a given resource[1] can be used by at most one task. The tasks of a job have to be processed in an order given as input. In the case of the MCJP, each task has a discrete consumption profile defined during its whole processing time. We write $conso_i^k$ the k^{th} value in the consumption profile of task i. At each time slot t of the horizon, the energy price is given by $price_t$. The cost of starting a task i at the time slot t is defined as $c_{i,t} = \sum_{k=0}^{p_i-1} conso_i^k * price_{t+k}$. The objective is to find a schedule minimizing the total cost of all tasks.

Constraint Programming Model For a job j, s_j (e_j) denotes the array of starting (completion) times of all tasks of job j, and s_j^i (e_j^i) the starting (completion) time of the i^{th} task of job j. Similarly s_r, e_r, p_r denotes the attributes (start/end/duration) of the tasks processed by resource r.

All tasks of a given job must be processed in sequence:

$$\forall j \in \mathcal{J}, \forall i \in [2..N], e_j^{i-1} \leq s_j^i$$

A machine can only process one task at a time. This is modeled with Unary resource constraints (also referred to as NoOverlap) [19]:

$$\forall\, r \in \mathcal{R} \quad \textsc{Unary}(s_r, p_r, e_r)$$

The objective $O = \sum_{i \in \mathcal{T}} c_{i,s_i}$ can be modeled with a Sum and Element constraints [18]. We can strengthen the filtering with the RCAD constraint [17] on each resource and each job: we split a task with a processing time p_i into p_i consecutive chunks with a duration of 1 time slot. Each of these chunks is then an item to be produced and has a consumption given by the task consumption profile.

3 The OrderedDisjunctiveCost constraint

In the CP model given in the previous section, the reasoning of the propagators involved in the cost computation does not take into account the tasks in

[1] Also called *machine* or *disjunctive resource*.

a job must be run in sequence. Moreover, the tasks are assumed to be preemptive, weakening the filtering. To cope with that problem, we introduce the OrderedDisjunctiveCost constraint, a new global constraint to be used in problems such as the MCJP. It is formally defined hereafter and Section 4 describes an efficient way to filter domains when this constraint must hold.

Definition 1. *Let $T = \langle (s_1, p_1, e_1), \ldots, (s_n, p_n, e_n) \rangle$ be a sequence of n tasks that must be executed in the given order, within their time window and that cannot overlap in time ; H be an integer constant, that is the number of time slots (horizon) ; c be a $n \times H$ matrix defining the integer costs of starting tasks at a given time (i.e., $c_{i,t}$ is the cost of starting task i at the time slot t); Z be an integer variable representing the total cost; the constraint* OrderedDisjunctiveCost*(T, c, Z) imposes that*

$$\wedge \begin{cases} \textsc{Unary}(T) \\ e_{i-1} \leq s_i \quad \forall i \in [2..n] \\ \sum_{i=1}^{n} c_{i,s_i} \leq Z \end{cases} \tag{1}$$

Intuitively, the constraint ensures tasks in T never overlap in time and are run in sequence in the given order. More importantly, it forces the total cost Z to be at least the cost for scheduling tasks in T.

The Unary can be enforced by efficient propagators [19, 6]. We assume that the relations $s_i \in [r_i, d_i - p_i] \ \wedge \ s_i + p_i = e_i \ , \forall i \in [1..n]$ are maintained by the Unary constraint. The ordering of the tasks can be constrained with binary constraints or a precedence graph (e.g., [19]). The focus of this paper is on the computation of the last part of the constraint : $\sum_{i=1}^{n} c_{i,s_i} \leq Z$.

The next section explains a dynamic program to compute the exact minimum total cost and induced variable-value assignment costs.

4 Exact Minimum Cost of a Task Sequence

Consider a sequence T of n ordered tasks and a time horizon $H = d_n$. Let $f^*(i,t)$ be the optimal (forward) value of a schedule involving tasks $1, \ldots, i$ and such that i finishes at the latest at the time slot t :

$$f^*(i,t) = \min \left\{ \sum_{k=1}^{i} c_{k,s_k} \mid e_i \leq t \wedge e_{k-1} \leq s_k \ \forall k \in [2..i] \wedge \textsc{Unary}(T[1..k]) \right\}$$

f^* verifies:

$$f^*(i,t) = \begin{cases} 0 & i = 0, \forall t \in [1..H] \quad \text{(empty set)} \\ +\infty & \forall i \in [1..n], t \in [1..r_i + p_i[\quad \text{(too early)} \\ \min(f^*(i,t-1), f^*(i-1,t-p_i) + c_{i,t-p_i}) & \\ & \forall i \in [1,n], t \in [r_i + p_i..d_i] \quad \text{(in domain)} \\ f^*(i,d_i) & \forall i \in [1..n], t \in]d_i..H] \quad \text{(too late)} \end{cases}$$

The first case corresponds to an empty set of tasks. For a task i, the last three cases span over the time horizon: $[1..r_i + p_i[\cup [r_i + p_i..d_i] \cup]d_i..H]$. Clearly,

if the task i cannot be terminated before (or exactly) at the time slot t, $f^*(i,t)$ is infinite (second case). If the task i cannot exactly terminate on the time slot t because $t > d_i$, $f^*(i,t)$ is equal to the minimum cost if it is finished *at the latest* at its deadline (last case). For a given time slot t, if task i can actually be scheduled, there are two possibilities: it is either better to finish it earlier than t (and the value is already known), or to finish it exactly at t. Figure 1 provides a visual illustration for this case. The optimal value for the n tasks is $f^*(n,H)$ which can be computed in space and time complexity $\mathcal{O}(n \cdot H)$.

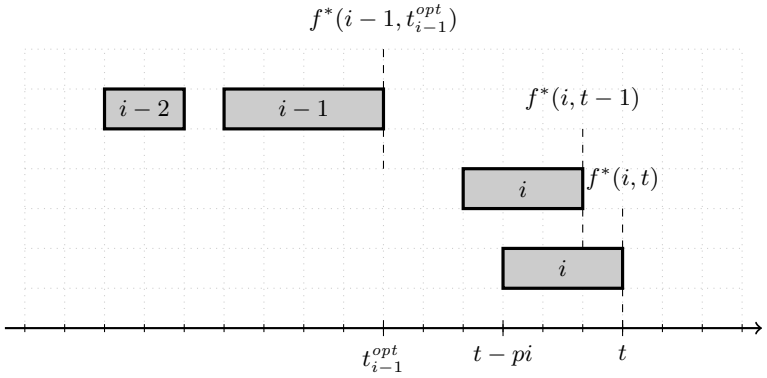

Fig. 1: Update rule of the dynamic program : is the schedule of tasks $1,\ldots,i$ finishing exactly on t cheaper than the optimum schedule finishing at most on $t-1$? That is, do we have $f^*(i-1,t_{i-1}^{opt}) + c_{i,t-p_i} = f^*(i-1,t-p_i) + c_{i,t-p_i} < f^*(i,t-1)$? t_{i-1}^{opt} is here the time slot that minimizes $f^*(i-1,t')\forall t' \leq t-p_i$.

Similarly one can compute the optimal (backward) value obtained by scheduling optimally the last $n-i+1$ tasks not earlier than t:

$$b^*(i,t) = \min\left\{\sum_{k=i}^{n} c_{k,s_k} \mid s_i \geq t \wedge e_{k-1} \leq s_k \ \forall k \in [i+1..n] \wedge \textsc{Unary}(T[i..n])\right\}$$

Let us define $Z^{s_i=t}$ the optimal cost of scheduling the n activities with the additional constraint that activity i is scheduled at the time t. This cost is computed in $\mathcal{O}(1)$ with the following formula assuming tables f^* and b^* are available:

$$Z^{s_i=t} = f^*(i-1,t) + c_{i,t} + b^*(i+1,t+p_i).$$

Example 1. Consider the set of tasks and the scheduling costs respectively defined in Figures 2a and 2b. The resulting tables of our dynamic program are given in Figures 2c and 2d. Consider task 2 in Figure 2c: $f^*(2,1) = f^*(2,2) = +\infty$ since task 2 can be ended at the earliest in 3 as $r_2 = 2$ and $p_2 = 1$. Also, $\forall t \in [6,8]$ $f^*(2,t) = f^*(2,d_2) = f^*(2,5) = 17$, which corresponds to the minimum cost schedule of the set of tasks $\{1,2\}$. Let us read the case where task

2 is scheduled correctly according to $D(s_2)$: scheduling task 2 at the earliest, we have $f^*(2,3) = \min(f^*(2,2), f^*(1,2) + c_{2,2}) = \min(+\infty, 14 + 12) = 26$. If we were to schedule it one time slot later (i.e., $s_2 = 3$), we would compute $f^*(2,4) = \min(f^*(2,3), f^*(1,3) + c_{2,3}) = \min(26, 29)$. In this case, ending task 2 earlier is cheaper and the value 26 is kept. The minimum cost of the whole schedule is given by $f^*(4,8) = b^*(1,0) = 25$. Once f^* and b^* are computed, one has access to all the induced variable-value assignment costs in $\mathcal{O}(1)$. For instance, $Z^{s_3=4} = f^*(2,4) + c_{3,4} + b^*(4, 4 + p_3) = 26 + 4 + 1 = 31$.

i	r_i	p_i	d_i
1	0	2	4
2	2	1	5
3	3	2	7
4	5	1	8

(a) Parameters of 4 tasks.

i \ t	0	1	2	3	4	5	6	7
1	14	20	23	13	5	5	14	14
2	3	12	12	15	3	3	3	12
3	7	16	17	16	4	4	7	7
4	1	4	4	5	1	1	1	4

(b) Starting costs.

i \ t	1	2	3	4	5	6	7	8
1	∞	14	14	14	14	14	14	14
2	∞	∞	26	26	17	17	17	17
3	∞	∞	∞	∞	42	30	21	21
4	∞	∞	∞	∞	∞	43	31	25

(c) Forward table f^*.

i \ t	0	1	2	3	4	5	6	7
1	25	31	34	∞	∞	∞	∞	∞
1	11	11	11	11	11	∞	∞	∞
1	5	5	5	5	5	8	∞	∞
1	1	1	1	1	1	1	1	4

(d) Backward table b^*.

Fig. 2: Input of Example 1 and resulting f^* and b^* tables.

Having computed the f^* and b^* functions, one can achieve Domain-Consistency using the rule:

$$\forall i \in T\ , \forall t \in D(s_i) : Z^{s_i=t} > \overline{Z} \implies s_i \neq t \tag{DC}$$

or Bound-Consistency[2] with the rule:

$$\forall i \in T\ , \forall t \in \{\underline{s_i}, \overline{s_i}\} : Z^{s_i=t} > \overline{Z} \implies s_i \neq t \tag{BC}$$

5 Experimentations

We conducted experiments on the MCJP. The model given in Section 2, written *RCAD*, is the baseline model against which we compared our approach. We extended it with bound-consistent and domain-consistent propagators on each job.

[2] Strictly speaking, Bound(D)-Consistency (see [2]).

Those models are referred to as $RCAD \cup DP_{BC}$ and $RCAD \cup DP_{DC}$. Our experiments were performed with the *OscaR* solver [10], AMD Opteron processors (2.7 GHz), the Java Runtime Environment 8, and a memory limit of 4 GB.

We used the *Replay Evaluation* framework [16]: it consists in generating a search tree using the *baseline* model (here $RCAD$) and traversing it with the different models in order to get metrics such as solving time and number of nodes. This evaluation allows ensuring that any gain in time/number of nodes can be attributed to additional propagation solely, and not to the dynamic search heuristic.

The search tree generation lasted 300 seconds. We used the conflict ordering search [7], an efficient search heuristic for scheduling problems. The initial variable choice heuristic is to branch on the variable with the smallest domain. For the value heuristic, we first branch on the available starting time minimizing the cost of the task, and we remove that value on the right branch.

We generated realistic instances (available here[3]) by using real historical electricity prices on the EU market. Tasks duration varies from 5 to 15, and consumptions within consumption profiles range between 1 and 20. The horizon H of an instance amounts to 105% of the minimum makespan of the instance regardless of the costs. The rationale is that industries are interested in allowing a bit of flexibility to their current schedule (which minimizes the makespan) in order to obtain energy costs savings. We generated two benchmarks of 100 instances with 10 resources, respectively with 5 and 10 jobs.

Results We provide the results as *performance profiles* [4, 15]. In the case of time, the performance profile of a model m is defined by:

$$F_m(\tau) = \frac{1}{|\mathcal{I}|} \left| \left\{ i \in \mathcal{I} : \frac{time_{replay}(m,i)}{\min_{m' \in M} time_{replay}(m',i)} \leq \tau \right\} \right|$$

where $\mathcal{I}$ is the set of considered instances and M is the set of all models.

Figure 3 provides the profiles for the time and the number of nodes metrics on the instances with 5 jobs. The time profiles first illustrate that $RCAD$ is always slower than our approach (as $F_{RCAD}(1) = 0$). Moreover, it is slower by a factor of at least 64 as compared with our approach for $\sim 45\%$ of the instances ($F_{RCAD}(64) \simeq 0.55$). Interestingly, the bound-consistent and domain-consistent filtering we introduce have similar time performances. Reading the profiles on the number of nodes, one can understand where the speedup comes from: the number of nodes is drastically reduced by our approach. Moreover, our domain-consistent propagator does not prune much more than our bound-consistent propagator. This explains the similar performances from a time perspective.

Let us briefly consider instances with 10 jobs with Figure 4: overall, the solving time improvement is less impressive (yet we are almost always faster). But this comes from a weaker pruning gain, rather than the fact that our algorithm does not scale to a larger horizon and number of tasks.

[3] Instances will be made available on CSPLib upon acceptance of the paper.

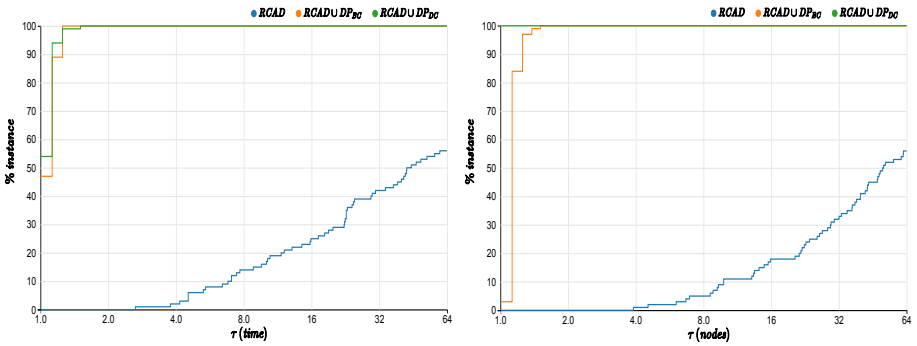

Fig. 3: Performance profiles for instances with 5 jobs and 10 machines.

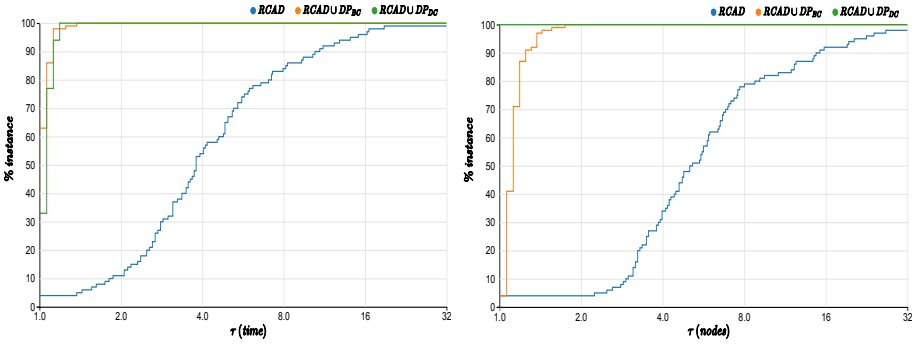

Fig. 4: Performance profiles for instances with 10 jobs and 10 machines.

6 Conclusion and Future Work

This paper considered scheduling problems where sets of tasks are known to be run in-sequence and scheduling tasks implies a cost depending on their starting time. In particular, we introduced the Minimum Cost JobShop Problem, an extension of the JopShop problem. We defined a new global constraint, OrderedDisjunctiveCost, that can be used in these kinds of problems. We described a dynamic program to compute the exact cost of a sequence of tasks and the induced minimum costs for all the variable-value assignments. This is the core of our filtering algorithm for this constraint. Our results on the MCJP reveal our approach significantly improves the state-of-the-art in CP.

This work is only the beginning of the story : a total order for a set of tasks is not always enforced in the definition of the problem. But scheduling propagators are often able to infer an order during search (e.g., tasks running on disjunctive resources, transitive closure on the precedence graph). We would like to extend our work so that it can be used for such settings. Moreover, we would like to extend the approach when tasks can overlap in time in a limited manner (e.g., for cumulative resources).

References

1. Baptiste, P., Le Pape, C., Nuijten, W.: Constraint-based scheduling: applying constraint programming to scheduling problems, vol. 39. Springer Science & Business Media (2012)
2. Bessiere, C.: Constraint propagation. Handbook of constraint programming pp. 85–134 (2006)
3. Dejemeppe, C., Devolder, O., Lecomte, V., Schaus, P.: Forward-checking filtering for nested cardinality constraints: Application to an energy cost-aware production planning problem for tissue manufacturing. In: International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming. pp. 108–124. Springer (2016)
4. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Mathematical programming 91(2), 201–213 (2002)
5. Ducomman, S., Cambazard, H., Penz, B.: Alternative filtering for the weighted circuit constraint: Comparing lower bounds for the TSP and solving TSPTW. In: AAAI Conference on Artificial Intelligence (2016)
6. Fahimi, H., Ouellet, Y., Quimper, C.G.: Linear-time filtering algorithms for the disjunctive constraint and a quadratic filtering algorithm for the cumulative not-first not-last. Constraints (Apr 2018)
7. Gay, S., Hartert, R., Lecoutre, C., Schaus, P.: Conflict ordering search for scheduling problems. In: International conference on principles and practice of constraint programming. pp. 140–148. Springer (2015)
8. Gay, S., Schaus, P., De Smedt, V.: Continuous casting scheduling with constraint programming. In: International Conference on Principles and Practice of Constraint Programming. pp. 831–845. Springer (2014)
9. Laborie, P., Rogerie, J., Shaw, P., Vilím, P.: Ibm ilog cp optimizer for scheduling. Constraints pp. 1–41 (2018)
10. OscaR Team: OscaR: Scala in OR (2012), available from `https://bitbucket.org/oscarlib/oscar`
11. Régin, J.C.: Cost-based arc consistency for global cardinality constraints. Constraints 7(3-4), 387–405 (2002)
12. Sellmann, M.: An arc-consistency algorithm for the minimum weight all different constraint. In: International Conference on Principles and Practice of Constraint Programming. pp. 744–749. Springer (2002)
13. Simonis, H., Hadzic, T.: A family of resource constraints for energy cost aware scheduling. In: Third International Workshop on Constraint Reasoning and Optimization for Computational Sustainability, St. Andrews, Scotland, UK (2010)
14. Simonis, H., Hadzic, T.: A resource cost aware cumulative. In: Recent Advances in Constraints: 14th Annual ERCIM International Workshop on Constraint Solving and Constraint Logic Programming, CSCLP 2009, Barcelona, Spain, June 15-17, 2009, Revised Selected Papers. pp. 76–89. Springer (2011)
15. Van Cauwelaert, S., Lombardi, M., Schaus, P.: A visual web tool to perform what-if analysis of optimization approaches. Tech. rep., UCLouvain (2016)
16. Van Cauwelaert, S., Lombardi, M., Schaus, P.: How efficient is a global constraint in practice? Constraints 23(1), 87–122 (2018)
17. Van Cauwelaert, S., Schaus, P.: Efficient filtering for the resource-cost alldifferent constraint. Constraints 22(4), 493–511 (2017)
18. Van Hentenryck, P., Carillon, J.P.: Generality versus specificity: An experience with ai and or techniques. In: AAAI. pp. 660–664 (1988)

19. Vilım, P.: Global constraints in scheduling. Ph.D. thesis, PhD thesis, Charles University in Prague, Faculty of Mathematics and Physics, Department of Theoretical Computer Science and Mathematical Logic, KTIML MFF, Universita Karlova, Malostranské námestı 2/25, 118 00 Praha 1, Czech Republic (2007)
20. Wamba, G.M., Beldiceanu, N.: The taskintersection constraint. In: International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming. pp. 246–261. Springer (2016)
21. Wüstenhagen, R., Bilharz, M.: Green energy market development in germany: effective public policy and emerging customer demand. Energy policy 34(13), 1681–1696 (2006)