Sequencing and counting with the multicost-regular constraint

Julien Menana, Sophie Demassey

Sequencing and Counting with the `multicost-regular` Constraint

Julien Menana and Sophie Demassey

École des Mines de Nantes, LINA CNRS UMR 6241, F-44307 Nantes, France.
{julien.menana,sophie.demassey}@emn.fr

Abstract. This paper introduces a global constraint encapsulating a `regular` constraint together with several cumulative costs. It is motivated in the context of personnel scheduling problems, where a schedule meets patterns and occurrence requirements which are intricately bound. The optimization problem underlying the `multicost-regular` constraint is NP-hard but it admits an efficient Lagrangian relaxation. Hence, we propose a filtering based on this relaxation. The expressiveness and the efficiency of this new constraint is experimented on personnel scheduling benchmark instances with standard work regulations. The comparative empirical results show how `multicost-regular` can significantly outperform a decomposed model with `regular` and `global-cardinality` constraints.

1 Introduction

Many combinatorial decision problems involve the simultaneous action of sequencing and counting objects, especially in the large class of routing and scheduling problems. In routing, a vehicle visits a sequence of locations following a path in the road network according to some numerical requirements on the whole travelling distance, the time spent, or the vehicle capacity. If only one numerical attribute is specified, finding a route is to solve a shortest/longest path problem. For several attributes, the problem – a Resource Constrained Shortest/Longest Path Problem (RCSPP) – becomes NP-hard. All these numerical requirements may drastically restrict the set of paths in the network which correspond to the actual valid routes. Hence, it is much more efficient to take these requirements into account throughout the search of a path, rather than each separately. Personnel scheduling problems can be treated analogously. Planning a worker schedule is to sequence activities (or shifts) over a time horizon according to many various work regulations, as for example: "a working night is followed by a free morning", "a night shift costs twice as musch as a day shift", "at least 10 days off a month", etc. Hence, a schedule meets both structural requirements – defined as allowed patterns of activities – and numerical requirements – defined as assignment costs or counters – which are intricately bounds. Modelling these requirements individually is itself a hard task, for which the expressiveness and the flexibility of Constraint Programming (CP) is recognized. Modelling these requirements efficiently is still a harder task as it means to aggregate all

of them in order to process this set of tied requirements as a whole. By introducing the `regular` global constraint, Pesant [1] has proposed an elegant and efficient way to model and to enforce all the pattern requirements together. The allowed patterns are gathered in an acyclic digraph whose paths coincide with the valid sequences of activities. This approach was later extended to optimization constraints `soft-regular` [2] and `cost-regular` [3] for enforcing bounds on the global cost – a violation cost or any financial cost – of the sequence of assignments. The underlying problem is now to compute shortest and longest paths in the acyclic graph of patterns. The `cost-regular` constraint was successfully applied to solve real-world personnel scheduling problems under a CP-based column-generation approach [3]. Nevertheless, the authors complained about the weak interaction in their CP model between the `cost-regular` constraint and an external `global-cardinality` used for modelling occurrence requirements. Actually, with such a decomposition, the support graph of `cost-regular` maintains many paths which do not satisfy the cardinality constraints. In this paper, we still generalize this approach for handling several cost attributes within one global constraint `multicost-regular`. Such a constraint allows to reason simultaneously on the sequencing and counting requirements occurring in personnel scheduling problems. As mentioned above, the underlying optimization problem is a RCSPP and it remains NP-hard even when the graph is acyclic. Hence, the filtering algorithm we present achieves a relaxed level of consistency. It is based on the Lagrangian relaxation of the RCSPP following the principle by Sellmann [4] for Lagrangian relaxation-based filtering. Our implementation of `multicost-regular` is available in the distribution of the open-source CP solver *CHOCO* [1].

The paper is organized as follows. In Section 2, we present the class of `regular` constraints and provide a theoretical comparison between the path-finding approach of Pesant [1] and the decomposition-based approach of Beldiceanu et al. [5]. We introduce then the new constraint `multicost-regular`. In Section 3, we introduce the Lagrangian relaxation-based filtering algorithm. In Section 4, we describe a variety of standard work regulations and investigate a systematic way of building one instance of `multicost-regular` from a set of requirements. In Section 5, comparative empirical results on benchmark instances of personnel scheduling problems are given. They show how `multicost-regular` can significantly outperform a decomposed model with `regular` and `global-cardinality` constraints.

2 Regular Language Membership Constraints

In this section, we recall the definition of the `regular` constraint and report on related work, before introducing `multicost-regular`. First, we recall basic notions of automata theory and introduce notations used throughout this paper:

We consider a non empty set Σ called the *alphabet*. Elements of Σ are called *symbols*, sequences of symbols are called *words*, and sets of words are called

[1] `http://choco.emn.fr/`

languages over Σ. An *automaton* Π is a directed multigraph (Q, Δ) whose arcs are labelled by the symbols of an alphabet Σ, and where two non-empty subsets of vertices I and A are distinguished. The set Q of vertices is called the set of *states* of Π, I is the set of *initial states*, and A is the set of *accepting states*. The non-empty set $\Delta \subseteq Q \times \Sigma \times Q$ of arcs is called the set of *transitions* of Π. A word in Σ is said to be *accepted* by Π if it is the sequence of the arc labels of a path from an initial state to an accepting state in Π. Automaton Π is a *deterministic finite automaton (DFA)* if Δ is finite and if it has only one initial state ($I = \{s\}$) and no two transitions sharing the same initial extremity and the same label. The language accepted by a FA is a *regular language*.

2.1 Path-Finding and Decomposition: Two Approaches for `regular`

The `regular` language membership constraint was introduced by Pesant in [1]. Given a sequence $X = (x_1, x_2, ..., x_n)$ of finite domain variables and a deterministic finite automaton $\Pi = (Q, \Sigma, \Delta, \{s\}, A)$, the constraint `regular`$(X, \Pi)$ holds iff X is a word of length n over Σ accepted by DFA Π. By definition, the solutions of `regular`(X, Π) are in one-to-one correspondance with the paths of exactly n arcs connecting s to a vertex in A in the directed multigraph Π. Let $\delta_i \in \Delta$ denote the set of transitions that appears as the i-th arc of such a path, then a value for x_i is consistent iff δ_i contains a transition labelled by this value.

Coincidently, Pesant [1] and Beldiceanu et al [5] introduce two orthogonal approaches to achieve GAC on `regular` (see Figure 1). The approach proposed by Pesant [1] is to unfold Π as an acyclic DFA Π_n which accepts only the words of length n. By construction, Π_n is a layered multigraph with state s in layer 0 (the source), the accepting states A in layer n (the sinks), and where the set of arcs in any layer i coincides with δ_i. A breadth-first search allows to maintain the coherence between Π_n and the variable domains by pruning the arcs in δ_i whose labels are not in the domain of x_i, then by pruning the vertices and arcs which are not connected to a source and to a sink. In Beldiceanu et al [5], a `regular` is decomposed as n tuple constraints for modelling the sets $\delta_1, \delta_2, \ldots, \delta_n$. The decomposition introduces state variables $q_0 \in \{s\}$, $q_1, \ldots, q_{n-1} \in Q$, $q_n \in A$ and uses triplet relations defined in extension to enforce GAC on the transition constraints $(q_{i-1}, x_i, q_i) \in \delta_i$. Such a constraint network being Berge-acyclic, enforcing AC on the decomposition achieves GAC on `regular`.

In the first approach, a specialized algorithm is defined to maintain all the support paths, while in the second approach, the transitions are modeled with tuple constraints which are directly propagated by the CP solver. The two approaches are orthogonal. Actually, the second model may mimic the specialized algorithm depending on the chosen propagation.

If we assume w.l.o.g. that Σ is the union of the variable domains, then the initial run of Pesant's algorithm for the construction of Π_n is performed in $O(n|\Delta|)$ time and space (with $\Delta \leq |Q||\Sigma|$ if Π is a DFA). Incremental filtering is performed with the same worst-case complexity with a forward/backward traversal of Π_n. Actually, the complexity of the algorithm relies more on the size $|\Delta_n|$ of the unfolded automaton Π_n rather than on the size $|\Delta|$ of the specified

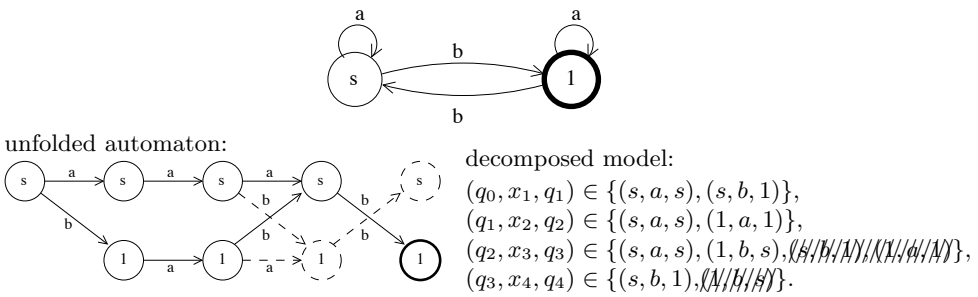


Fig. 1. Consider the DFA depicted above applied to $X \in \{a, b\} \times \{a\} \times \{a, b\} \times \{b\}$. The unfolded automaton of `regular` is depicted on the left and the decomposed model on the right. The dashed transitions are discarded in both models.

automaton Π. Note for instance that when the specified automaton Π accepts only words of length n then it is already unfolded ($\Pi = \Pi_n$) and the first run of the algorithm is in $O(|\Delta|)$. In practice, as in our experiments (Section 5), Π_n can even be much smaller than Π, meaning that many accepting states in Π cannot be reached in exactly n transitions. The incremental filtering is performed in $O(|\Delta_n|)$ time with, in such a case, $|\Delta_n| \ll n|\Delta|$.

`regular` is a very expressive constraint. It is useful to model pattern constraints arising in many planning problems, but also to reformulate other global constraints [5] or to model tuples defined in extension. An other application of `regular` is to model a sliding constraint: recently, Bessière et al. [6] have introduced the `slide` meta constraint. In its more general form, `slide` takes as arguments a matrix of variables Y of size $n \times p$ and a constraint C of arity pk with $k \leq n$. `slide`(Y, C) holds if and only of $C(y^1_{i+1}, \ldots, y^p_{i+1}, \ldots, y^1_{i+k}, \ldots, y^p_{i+k})$ holds for $0 \leq i \leq n-k$. Using the decomposition proposed in [5], `regular`(X, Π) can be reformulated as `slide`$([Q, X], C_\Delta)$, where Q is the sequence of state variables and C_Δ is the transition constraint $C_\Delta(q, x, q', x') \equiv (q, x, q') \in \Delta$. Conversely [6], a `slide` constraint can be reformulated as a `regular` but it may require to enumerate all valid tuples for C. This reformulation can however be useful in the context of planning (especially for car sequencing) to model a sliding cardinality constraint also known as `sequence`. Even if powerful specialized algorithms exist for this constraint (see e.g. [7]), the automaton resulting from the reformulation can be integrated with other pattern requirements as we will show in Section 4. Finally, one should notice the work (see e.g. [8]) related to context-free grammar constraints. Though, most of the rules encountered in personnel scheduling can be described using regular languages.

2.2 Maintaining Patterns with Cumulative Costs and Cardinalities

Personnel scheduling problems are usually defined as optimization problems. Most often, the criterion to optimize is a *cumulative cost*, i.e. the sum of costs associated to each assignment of a worker to a given activity at a given time.

Such a cost has several meanings: it can model a financial cost, a preference, or a value occurrence. Now, designing a valid schedule for one worker is to enforce the sequence of assignments to comply with a given pattern while ensuring that the total cost of the assignments is bounded. This can be specified by means of a `cost-regular` constraint [3]. Given $c = (c_{ia})_{i \in [1..n] \times a \in \Sigma}$ a matrix of real assignment costs and $z \in [\underline{z}, \overline{z}]$ a bounded variable ($\underline{z}, \overline{z} \in \mathbb{R}$), `cost-regular`$(X, z, \Pi, c)$ holds iff `regular`(X, Π) holds and $\sum_{i=1}^{n} c_{ix_i} = z$. Note that it has the `knapsack` constraint [9] as a special case and that, unless $P = NP$, one can enforce GAC on a `knapsack` constraint at best in pseudo-polynomial time, i.e. the run time is polynomial in the values of the bounds of z. As a consequence, enforcing GAC on `cost-regular` is NP-hard.

The definition of `cost-regular` reveals a natural decomposition as a `regular` constraint channeled to a `knapsack` constraint. Actually, it is equivalent to the decomposition proposed by Beldiceanu et al. [5] when dealing with one cumulative[2] cost: cost variables k_i are now associated to the previous state variables q_i, with $k_0 = 0$ and $k_n = z$, and several arithmetic and `element` constraints model the `knapsack` and channeling constraints. In short, this formulation can be rewritten as `slide`$([Q, X, K], C^c_\Delta)$, with $C^c_\Delta(q_{i-1}, x_{i-1}, k_{i-1}, q_i, x_i, k_i) \equiv (q_{i-1}, x_{i-1}, q_i) \in \Delta \wedge k_i = k_{i-1} + c_{ix_i}$. Depending on the size of the domains of the cost variables, GAC can be enforced on `knapsack` in reasonable time. However, even in this case, since the constraint hypergraph of the decomposed model is no longer Berge-acyclic but α-acyclic, one has to enforce pairwise-consistency on the shared variables – a pair (q_i, k_i) of state and cost variables – of the transition constraints in order to achieve GAC. A similar option proposed for `slide` [6] is to enforce AC on the dual encoding of the hypergraph of the C^c_Δ constraints, but again it requires to explicit all the support tuples and then, it may be of no practical use.

The filtering algorithm presented in [3] for `cost-regular` is a slight adaptation[3] of Pesant's algorithm for `regular`. It is based on the computation of shortest and longest paths in the unfolded graph Π_n valued by the transition costs. To each vertex (i, q) in any layer i of Π_n are associated two bounded cost variables k^-_{iq} and k^+_{iq} modelling the lengths of the paths respectively from layer 0 to (i, q) and from (i, q) to layer n. The cost variables can trivially be initialized during the construction of Π_n: k^-_{iq} in the forward phase and k^+_{iq} in the backward phase. The bounds of variable z are then pruned according to the condition $z \subseteq k^+_{0s}$. Conversely, an arc $((i-1, q), a, (i, q')) \in \delta_i$ can be removed whenever:

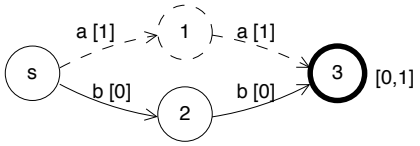
$$\underline{k^-_{(i-1)q}} + c_{ia} + \underline{k^+_{iq'}} > \overline{z} \quad \text{or} \quad \overline{k^-_{(i-1)q}} + c_{ia} + \overline{k^+_{iq'}} > \underline{z}.$$

As graph Π_n is acyclic, maintaining the cost variables, i.e. shortest and longest paths, can be performed by breadth-first traversal with the same time complexity $O(|\Delta_n|)$ than for maintaining the connexity of the graph in `regular`.

[2] The model in [5] can deal not only with sum but also with various arithmetic functions on costs, but no example of use is provided.

[3] Previously, the algorithm was partially – for minimization only – applied to the special case `soft-regular[hamming]` in [5] and in [2].

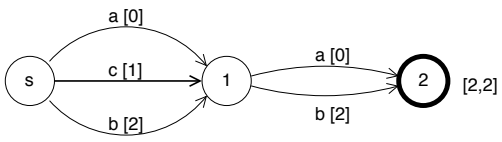
As said before, this algorithm achieves a hybrid level of consistency on `cost-regular`. As a matter of fact, it enforces a sort of pairwise-consistency on the decomposed model between each state variable and the bounds of the associated cost variable, according to the relation $q_i = (i, q) \iff k_i = k^-_{iq}$. Hence, it dominates the decomposed model `knapsack`$\wedge$`regular` when only Bound Consistency is enforced on the cost variables. Otherwise, if AC is enforced on `knapsack` then the two approaches are incomparable as show the two examples depicted in Figures 2 and 3.



$x_1 \in \{\not{a}, b\}, x_2 \in \{\not{a}, b\}, z \in [0, 1]$.

$(q_0, x_1, q_1, j_1) \in \{(s, a, 1, 1), (s, b, 2, 2)\}$,
`element`$(k_1, j_1, (k_0 + 1, k_0))$,
$(q_1, x_2, q_2, j_2) \in \{(1, a, 3, 1), (2, b, 3, 2)\}$,
`element`$(z, j_2, (k_1 + 1, k_1))$,
$q_0 \in \{s\}, q_1 \in \{1, 2\}, q_2 \in \{3\}$,
$k_0 \in \{0\}, k_1 \in \{0, 1\}$,
$x_1 \in \{a, b\}, x_2 \in \{a, b\}, z \in \{0, 1\}$.

Fig. 2. Consider the depicted DFA with costs in brackets applied to $X = (x_1, x_2) \in \{a, b\} \times \{a, b\}$ and $z \in [0, 1]$. The `cost-regular` algorithm (on the left) discards the dashed transitions and hence achieves GAC. The decomposed model (on the right) is arc-consistent but not globally consistent.



$x_1 \in \{a, b, c\}, x_2 \in \{a, b\}, z \in [2, 2]$.

$(x_1, j_1) \in \{(a, 1), (\not{a}, 2), (b, 3)\}$,
`element`$(k_1, j_1, (k_0, \not{k_0}, k_0 + 2))$,
$(x_2, j_2) \in \{(a, 1), (b, 2)\}$,
`element`$(z, j_2, (k_1, k_1 + 2))$,
$k_0 \in \{0\}, k_1 \in \{0, \not{1}, 2\}$,
$x_1 \in \{a, b, \not{c}\}, x_2 \in \{a, b\}, z \in \{2\}$.

Fig. 3. Consider now the depicted DFA applied to $X = (x_1, x_2) \in \{a, b, c\} \times \{a, b\}$ and $z \in [2, 2]$. Enforcing AC on the decomposed model (on the right) achieves GAC. The cost-regular algorithm (on the left) does not achieve GAC since the minimum and maximum paths traversing arc $x_1 = c$ are consistent with the bounds on z.

2.3 The `multicost-regular` Constraint

A natural generalization of `cost-regular` is to handle several cumulative costs: given a vector $Z = (z^0, ..., z^R)$ of bounded variables and $c = (c^r_{ia})^{r \in [0..R]}_{i \in [1..n], a \in \Sigma}$ a matrix of assignment costs, `multicost-regular`(X, Z, Π, c) holds if and only if `regular`(X, Π) holds and $\sum_{i=1}^{n} c^r_{ix_i} = z^r$ for all $0 \leq r \leq R$. Such a generalization has an important motivation in the context of personnel scheduling. Actually, apart a financial cost and pattern restrictions, an individual schedule

is usually subject to a `global-cardinality` constraint bounding the number of occurrences of each value in the sequence. These bounds can drastically restrict the language on which the schedule is defined. Hence, it could be convenient to tackle them within the `regular` constraint in order to reduce the support graph. As a generalization of `cost-regular` or of the `global-sequencing` constraint [10], we cannot hope to achieve GAC in polynomial time here. Note that the model by Beldiceanu et al [5] – and similarly the `slide` constraint – was also proposed for dealing with several costs but again, it amounts to decompose as a `regular` constraint channeled with one `knapsack` constraint for each cost.

Hence, we ought to exploit the structure of the support graph of Π_n to get a good relaxed propagation for `multicost-regular`. The optimization problems underlying `cost-regular` were shortest and longest path problems in Π_n. The optimization problems underlying `multicost-regular` are now the Resource Constrained Shortest and Longest Path Problems (RCSPP and RCLPP) in Π_n. The RCSPP (resp. RCLPP) is to find the shortest (resp. longest) path between a source and a sink in a valued directed graph, such that the quantities of resources accumulated on the arcs do not exceed some limits. Even with one resource on acyclic digraphs, this problem is known to be NP-hard[11]. Two approaches are most often used to solve RCSPP [11]: dynamic programming and Lagrangian relaxation. Dynamic programming-based methods extend the usual short path algorithms by recording the costs over every dimension at each node of the graph. As in `cost-regular`, this could easily be adapted for filtering by converting these cost labels as cost variables but it would make the algorithm memory expensive. Instead, we investigate a Lagrangian relaxation approach, which can also easily be adapted for filtering from the `cost-regular` algorithm without memory overhead.

3 A Lagrangian Relaxation-Based Filtering Algorithm

Sellmann [4] laid the foundation for using the Lagrangian relaxation of a linear program to provide a cost-based filtering for a minimization or maximization constraint. We apply this principle to the RCSPP/RCLPP for filtering `multicost-regular`. The resulting algorithm is a simple iterative scheme where filtering is performed by `cost-regular` on Π_n for different aggregated cost functions. In this section, we present the usual Lagrangian relaxation model for the RCSPP and explain how to solve it using a subgradient algorithm. Then, we show how to adapt it for filtering `multicost-regular`.

Lagrangian Relaxation for the RCSPP. Consider a directed graph $G = (V, E, c)$ with source s and sink t, and resources $(\mathcal{R}_1, ..., \mathcal{R}_R)$. For each resource $1 \leq r \leq R$, let $\overline{z_r}$ (resp. $\underline{z_r}$) denote the maximum (resp. minimum[4]) capacity of a path over the resource r, and c_{ij}^r denote the consumption of resource r on

[4] in the original definition of RCSPP, there is no lower bound on the capacity: $\underline{z_r}$

arc $(i,j) \in E$. A binary linear programming formulation for the RCSPP is as follows:

$$\min \sum_{(i,j)\in E} c_{ij}x_{ij} \tag{1}$$

$$\text{s.t. } \underline{z^r} \leq \sum_{(i,j)\in E} c^r_{ij}x_{ij} \leq \overline{z^r} \qquad \forall r \in [1..R] \tag{2}$$

$$\sum_{j\in V} x_{ij} - \sum_{j\in V} x_{ji} = \begin{cases} 1 & \text{if } \ i = s, \\ -1 & \text{if } \ i = t, \\ 0 & \text{otherwise.} \end{cases} \qquad \forall i \in V \tag{3}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in E. \tag{4}$$

In this model, a binary decision variable x_{ij} defines whether arc (i,j) belongs to a solution path. Constraints (2) are the resource constraints and Constraints (3) are the usual path constraints.

Lagrangian relaxation consists in dropping "complicating constraints" and adding them to the objective function with a violation penalty cost $u \geq 0$, called the *Lagrangian multipliers*. The resulting program is called the Lagrangian subproblem with parameter u and it is a relaxation of the original problem. Solving the *Lagrangian dual* is to find the multipliers $u \geq 0$ which gives the best relaxation, i.e. the maximal lower bound.

The complicating constraints of the *RCSPP* are the $2R$ resource constraints (2). Indeed, relaxing these constraints leads to a shortest path problem, that can be solved in polynomial time. Let P denote the set of solutions $x \in \{0,1\}^E$ satisfying Constraints (3). P defines the set of paths from s to t in G. The Lagrangian subproblem with given multipliers $u = (u_-, u_+) \in \mathbb{R}^{2R}_+$ is:

$$SP(u): \quad f(u) = \min_{x\in P} cx + \sum_{r=1}^{R} u^r_+(c^r x - \overline{z^r}) - \sum_{r=1}^{R} u^r_-(c^r x - \underline{z^r}) \tag{5}$$

An optimal solution x^u for $SP(u)$ is then a shortest path in graph $G(u) = (V, E, c(u))$ where:

$$c(u) = c + \sum_{r=1}^{R}(u^r_+ - u^r_-)c^r, \kappa_u = \sum_{r=1}^{n}(u^r_-\underline{z^r} - u^r_+\overline{z^r}) \text{ and } f(u) = c(u)x^u + \kappa_u. \tag{6}$$

Solving the Lagrangian Dual. The Lagrangian dual problem is to find the best lower bound $f(u)$, i.e. to maximize the piecewise linear concave function f:

$$LD: \quad f_{LD} = \max_{u\in\mathbb{R}^{2R}_+} f(u) \tag{7}$$

Several algorithms exist to solve the Lagrangian dual. In our approach, we consider the subgradient algorithm [12] as it is rather easy to implement and it does not require the use of a linear solver. The subgradient algorithm iteratively solves

one subproblem $SP(u)$ for different values of u. Starting from an arbitrary value, the position u is updated at each iteration by moving in the direction of a supergradient Γ of f with a given step length μ: $u^{p+1} = max\{u^p + \mu_p \Gamma(u^p), 0\}$. There exist many ways to choose the step lengths for guaranteeing the convergence of the subgradient algorithm towards f_{LD} (see e.g. [13]). In our implementation, we use a standard step length $\mu_p = \mu_0 \epsilon^p$ with μ_0 and $\epsilon < 1$ "sufficiently" large (we have empirically fixed $\mu_0 = 10$ and $\epsilon = 0.8$). For the supergradient, solving $SP(u)$ returns an optimum solution $x^u \in P$ and $\Gamma(u)$ is computed as: $\Gamma(u) = ((c^r x^u - \overline{z^r})_{r\in[1..R]}, (\underline{z^r} - c^r x^u)_{r\in[1..R]})$.

From Lagrangian Relaxation to Filtering. The key idea of Lagrangian relaxation-based filtering, as stated in [4], is that if a value is proved to be inconsistent in at least one Lagrangian subproblem then it is inconsistent in the original problem:

Theorem 1. *(i) Let P be a minimization linear program with optimum value $f^* \leq +\infty$, $\overline{z} \leq +\infty$ be an upper bound for P, and $SP(u)$ be any Lagrangian subproblem of P, with optimum value $f(u)^* \leq +\infty$. If $f(u) > \overline{z}$ then $f^* > \overline{z}$.*
(ii) Let x be a variable of P and v a value in its domain. Consider $P_{x=v}$ (resp. $SP(u)_{x=v}$) the restriction of P (resp. $SP(u)$) to the set of solutions satisfying $x = v$ and let $f^*_{x=v} \leq +\infty$ (resp. $f(u)_{x=v} \leq +\infty$) its optimum value. If $f(u)_{x=v} > \overline{z}$ then $f^*_{x=v} > \overline{z}$.

Proof. Statement (i) of Theorem 1 is straightforward, since $SP(u)$ is a relaxation for P, then $f(u) \leq f^*$. Statement (ii) arises from (i) and from the fact that, adding a constraint $x = v$ within P and applying Lagrangian relaxation, or applying Lagrangian relaxation and then adding constraint $x = v$ to each subproblem, result in the same formulation.

The mapping between `multicost-regular`(X, Z, Π, c), with $|Z| = R+1$ and an instance of the RCSPP (resp. RCLPP) is as follows: We single out one cost variable, for instance z^0, and create R resources, one for each other cost variable. The graph $G = (\Pi_n, c^0)$ is considered. A feasible solution of the RCSPP (resp. RCLPP) is a path in Π_n from the source (in layer 0) to a sink (in layer n) that consumes on each resource $1 \leq r \leq R$ is at least $\underline{z^r}$ and at most $\overline{z^r}$. Furthermore, we want to enforce an upper bound $\overline{z^0}$ on the minimal value for the RCSPP (resp. a lower bound $\underline{z^0}$ on the maximal value for the RCLPP). The arcs of Π_n are in one-to-one correspondance with the binary variables in the linear model of these two instances.

Consider a Lagrangian subproblem $SP(u)$ of the RCSPP instance (the approach is symmetric for the maximization instance of RCLPP) . We show that a slight modification of the `cost-regular` algorithm allows to solve $SP(u)$ but also to prune arcs of Π_n according to Theorem 1 and to shrink the lower bound $\underline{z^0}$. The algorithm starts by updating the costs on the graph Π_n with $c^0(u)$, as defined in (6) and then by computing, at each node (i, q), the shortest path $\underline{k^-_{iq}}$ from layer 0 and the shortest path $\underline{k^+_{iq}}$ to layer n. We get the optimum value

$f(u) = \underline{k^+_{0s}} + \kappa_u$. As it is a lower bound for z^0, one can eventually update this lower bound as $\underline{z^0} = \max\{f(u), \underline{z^0}\}$. Then, by a traversal of Π_n, we remove each arc $((i-1,q),a,(i,q')) \in \delta_i$ such that $\underline{k^-_{(i-1)q}} + c^0(u)_{ia} + \underline{k^+_{iq'}} > \overline{z^0} - \kappa_u$.

The global filtering algorithm we developed for `multicost-regular` is as follows: starting from $u = 0$, a subgradient algorithm guides the choice of the Lagrangian subproblems to which the above cost-filtering algorithm is applied. The number of iterations for the subgradient algorithm is limited to 20 (it usually terminates far before). The subgradient algorithm is first applied to the minimization problem (RCSPP) then to the maximization problem (RCLPP). As a final step, we run the original `cost-regular` algorithm on each of the cost variables to shrink their bounds (by the way, it could deduce new arcs to filter, but it did not happen in our experiments). Note that due to the parameter dependancy of the subgradient algorithm, the propagation algorithm is not monotonic.

4 Modelling Personnel Scheduling Problems

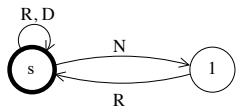
In this section we show how to model standard work regulations arising in Personnel Scheduling Problems (PSP) as one instance of the `multicost-regular` constraint. The purpose is to emphasize the ease of modelling with such a constraint and also to derive a systematic way of modelling PSP.

4.1 Standard work regulations

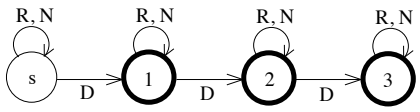
In PSP, many kinds of work regulations can be encountered, however, we can categorize most of them as rules enforcing either regular patterns, fixed cardinalities or sliding cardinalities.

To illustrate those categories, we consider a 7 days schedule and 3 activities: night shift (N), day shift (D) or rest shift (R). For example: R|R|D|D|N|R|D

Regular patterns can be modelled directly as a DFA. For instance the rule "a night shift is followed by a rest" is depicted in Figure 4 (A). The rules can either be given as forbidden patterns or allowed patterns. In the first case, one just need to build the complement automaton.



(A) a regular pattern example.



(B) cardinality rule example.

Fig. 4. Examples of automata representing work regulations.

Fixed cardinality rules bound the number of occurrences of an activity or a set of activities over a fixed subsequence of time slots. Such a rule can be modelled within an automaton or using counters. For example, the rule "at least 1 and at most 3 day shifts each week" can easily be modelled as the DFA depicted in Figure 4 (B). Taking a look at this automaton, we can see the initial state has been split into 3 different states that represent the maximum number of D transitions that can be taken. Such a formulation can be an issue when the maximum occurrence number increases. In this case, using a counter is more suitable, as we only need to create a new cost variable $z^r \in [1, 3]$ with $c^r_{ij} = 1 \iff 1 \leq i \leq 7$ and $j = D$. More generally, one can also encounter cardinality rules over patterns. This also can be managed by means of a cost. One has to isolate the pattern within the automaton describing all the feasible schedules, then to price transitions entering it to 1.

Sliding constraints can be modelled as a DFA using the reformulation stated in [6]. However, the width of the sliding sequence should not be too large as the reformulation requires to explicit all the feasible tuples of the constraint to slide. This is often the case in PSP or also in car sequencing problems.

4.2 Systematic `multicost-regular` Generation

A formalism to describe Personnel Scheduling Problems has been proposed in [14]. The set of predefined XML markups allows to specify a large scope of PSP. In order to automatically generate a CP model based on `multicost-regular` from such specifications, we developed a framework capable of interpreting those XML files. In a first step, we bounded each markup associated to a work regulation to one of the 3 categories described above. Hence, for each rule of a given PSP instance, we automatically generate either an automaton or a counter depending on the rule category. For instance, the forbidden pattern "no day shift just after a night shift" is defined in the xml file as

```
<Pattern weight="1350"><Shift>N</Shift><Shift>D</Shift></Pattern>
```

and is automatically turned into its equivalent regular expression $(D|N|R) * ND(D|N|R)*$. We use a java library for automata[5] in order to create a DFA from a regular expression and to operate on the set of generated DFA. We use the opposite, the intersection and the minimization operations to build an unique DFA. Once the DFA is built, we treat the rules that engender counters, and generate a `multicost-regular` instance for each employee. Last, we treat the transversal constraints and include them in the CP model. For example, cover requirements are turned into `global-cardinality` constraints. Note that we were not able to deal with two kinds of specifications: some rule violation penalties that the `multicost-regular` cannot model and the pattern cardinality rules that we do not yet know how to automatize the reformulation.

[5] http://www.brics.dk/automaton/

4.3 Two Personnel Scheduling Cases

We first tackled the *GPost* [14] problem. This PSP consists in building a valid schedule of 28 days for eight employees. Each day, an employee has to be assigned to a Day, Night, or Rest Shift. Each employee is bound to a (Fulltime or Part-time) contract defining regular pattern and cardinality rules. Regular pattern rules are: "free days period should last at least two days", "consecutive working week-ends are limited" and "given shift sequences are not allowed". Using the automatic modelling method we presented earlier, we build a DFA for each kind of contract. Cardinality rules are: "a maximum number of worked days in the 28 days period is to be worked", "the amount of certain shifts in a schedule is limited" and "the number of working days per week is bounded". Cover requirements and employee availabilities are also modelled. The softness specification on rules has been ignored as well as the first pattern rule to avoid infeasibility.

The second case study is based on the generated benchmark set brought by Demassey et al. [3]. The work regulations arise from a real-world personnel scheduling problem. The goal is to build only one schedule for a day consisting of 96 fifteen minutes time slots. Each slot is assigned either a working activity, a break, a lunch or a rest. Each possible assignment carries a given cost. The purpose is to find a schedule of minimum cost meeting all the work regulations. As for the previous PSP, we can identify regular pattern work regulations: "A working activity lasts at least 1 hour", "Different work activities are separated by a break or a lunch", "Break, lunch and rest shifts cannot be consecutives", "Rest shifts are at the beginning or at the end of the day", and "A break lasts 15 minutes". And also fixed cardinality regulations with: "At least 1 and at most 2 breaks a day", "At most one lunch a day" and "Between 3 and 8 hours of work activities a day". In addition to those work regulations, some activities are not allowed to be performed during some period. These rules are trivial to model with unary constraints.

5 Experiments

Experimentations were run on an Intel Core 2 Duo 2Ghz processor with 2048MB of RAM running OS X. The two PSP problems were solved using the Java constraint library *CHOCO* with default value selection heuristics – min value.

5.1 On the Size of the Automaton

As explained in Section 2.1, the filtering algorithm complexity of the `regular` constraints depends on the size of the specified automaton. Thus it would seem natural that processing a big automaton is not a good idea. However, practical results points out two important facts. First of all, the operations we run for automatically building a DFA from several rules tends to generate partially unfolded DFA (by intersection) and to reduce the number of redundant states which lie in the same layers (by minimization). Hence, the unfolded automaton

generated during the forward phase at the initialization of the constraint can even be smaller than the specified automaton Π. Secondly, pruning during the backward phase may produce an even smaller automaton Π_n as many accepting states cannot be reached in a given number n of transitions. Table 1 shows the number of nodes and arcs of the different automata during the construction of the `multicost-regular` constraints for the *GPost* problem: the sum of the DFAs generated for each rule, the DFA Π after intersection and minimization, the unfolded DFA after the forward and backward phases.

Contract	Count	sum of DFAs	Π	Forward	Backward (Π_n)
Fulltime	# Nodes	5782	682	411	230
	# Arcs	40402	4768	1191	400
Parttime	# Nodes	4401	385	791	421
	# Arcs	30729	2689	2280	681

Table 1. Illustration of graph reduction during presolving.

5.2 Comparative Experiments

The previous section showed the ease of modelling with `multicost-regular`. However, there would be no point in defining such a constraint if the solving was badly impacted. We then conduct experiments for comparing our algorithm with a decomposed model consisting of a `regular` (or `cost-regular` for optimization) channeled to a `global-cardinality` constraint (`gcc`).

Table 2 presents the computational results on the *GPost* instance. The models include 8 `multicost-regular` or 8 `regular` and `gcc` (for each employee) bound together by 28 transversal `gcc` (for each day). In the Table, the first row corresponds to the problem without the sliding rule over the maximum number of consecutive working week-ends. In the second row, this constraint was included. We tried various variable selection heuristics but found out assigning variables along the days gave the best results as it allows the constraint solver to deal with the transversal `gcc` more efficiently. Both models lead to the same

	`multicost-regular`		`regular` $\wedge$ `gcc`	
WE regulation	Time (s)	# Fails	Time (s)	# Fails
no	1.94	24	12.6	68035
yes	16.0	1576	449.2	2867381

Table 2. *GPost* problem results

solutions. Actually, the average time spent on each node is much bigger using `multicost-regular`. However, due to better filtering capabilities, the size of the search tree and the runtime to find a feasible solution are significantly decreased.

Our second experiment tested the scalability of `multicost-regular` (MCR) against `cost-regular^gcc` (CR) on the optimization problem defined in Section 4.3. The models do not contain any other constraint. However the decomposed model CR requires additional channeling variables. Table 3 presents the results on a benchmark set made of 110 instances. The number n of working activities varies between 1 and 50. The assignment costs were randomly generated. We tested different variable selection heuristics and kept the best one for each model. Note that the results of the CR model are more impacted by the heuristic.

The first columns in the table show that with the MCR model, we were able to solve all instances (Column #) in less than 15 seconds for the biggest ones (Column t). The average number of backtracks (Column bt) remains stable and low as n increases. On the contrary the CR model is impacted a lot as shown in the next columns. Indeed, as the initial underlying graph becomes bigger it contains more and more paths violating the cardinality constraints. Those paths are not discarded by `cost-regular`. Some instances with more than 8 activities could not be solved within the given 30 minutes (Column #). Considering only solved instances, the running time (t) and the number of backtracks (bt) are always much higher than the MCR model based results. Regarding unsolved instances, the best found solution within 30 minutes is rarely optimal (Column # opt), and the average gap (Column Δ) is up to 6% for 40 activities.

	`MCR` model			`CR` model					
	Solved			Solved			Time out		
n	#	t	bt	#	t	bt	#	Δ	# opt
1	10	0.6	49	10	1.1	292	0	-	-
2	10	0.8	54	10	2.4	539	0	-	-
4	10	1.5	65	10	13.5	1638	0	-	-
6	10	1.6	44	10	53.6	4283	0	-	-
8	10	2.1	51	9	209.2	5132	1	3.5%	0
10	10	2.4	58	7	283.5	6965	3	4.6%	0
15	10	3.8	59	6	283.9	4026	4	4.7%	1
20	10	4.9	49	6	311.8	4135	4	4.2%	1
30	10	6.9	51	1	313.0	4303	9	3.1%	1
40	10	13.4	68	0	-	-	10	6.1%	0
50	10	14.4	51	1	486.0	1406	9	5.0%	1

Table 3. Shift generation results

6 Conclusion

In this paper, we introduce the `multicost-regular` global constraint and provide a simple implementation of Lagrangian relaxation-based filtering for it. Experimentations on benchmark instances of personnel scheduling problems show

the efficiency and the scalability of this constraint compared to a decomposed model dealing with pattern requirements and cardinality requirements separately. Furthermore, we investigate a systematic way to build an instance of `multicost-regular` from a given set of standard work regulations. In future works, we ought to get a fully systematic system linked to the *CHOCO* solver for modelling and solving a larger variety of personnel scheduling and rostering problems.

Acknowledgements

We thank Mats Carlsson for pointing out the example illustrating the propagation issue of the `cost-regular` algorithm. We also thank Christian Schulte for its insightful comments on the paper and its numerous ideas to improve the constraint.

References

1. Pesant, G.: A regular language membership constraint for finite sequences of variables. In: Proceedings of CP'2004. (2004) 482–495
2. van Hoeve, W.J., Pesant, G., Rousseau, L.M.: On global warming: Flow-based soft global constraints. J. Heuristics **12**(4-5) (2006) 347–373
3. Demassey, S., Pesant, G., Rousseau, L.M.: A cost-regular based hybrid column generation approach. Constraints **11**(4) (2006) 315–333
4. Sellmann, M.: Theoretical foundations of CP-based lagrangian relaxation. Principles and Practice of Constraint Programming –CP 2004 (2004) 634–647
5. Beldiceanu, N., Carlsson, M., Debruyne, R., Petit, T.: Reformulation of Global Constraints Based on Constraint Checkers. Constraints **10**(3) (2005)
6. Bessière, C., Hebrard, E., Hnich, B., Kiziltan, Z., Quimper, C.G., Walsh, T.: Reformulating global constraints: The SLIDE and REGULAR constraints. In: SARA. (2007) 80–92
7. Maher, M., Narodytska, N., Quimper, C.G., Walsh, T.: Flow-Based Propagators for the *sequence* and Related Global Constraints. In: Proceedings of CP'2008. Volume 5202 of LNCS. (2008) 159–174
8. Kadioglu, S., Sellmann, M.: Efficient context-free grammar constraints. In: AAAI. (2008) 310–316
9. Trick, M.: A dynamic programming approach for consistency and propagation for knapsack constraints (2001)
10. Régin, J.C., Puget, J.F.: A filtering algorithm for global sequencing constraints. In: CP. (1997) 32–46
11. Handler, G., Zang, I.: A dual algorithm for the restricted shortest path problem. Networks **10** (1980) 293–310
12. Shor, N., Kiwiel, K., Ruszcayǹski, A.: Minimization methods for non-differentiable functions. Springer-Verlag New York, Inc. (1985)
13. Boyd, S., Xiao, L., Mutapcic, A.: Subgradient methods. lecture notes of EE392o, Stanford University, Autumn Quarter **2004** (2003)
14. Personnel Scheduling Data Sets and Benchmarks: `http://www.cs.nott.ac.uk/~tec/NRP/`