The Cyclic SEQUENCE Constraint

Nina Narodytska and **Toby Walsh**
University of New South Wales and NICTA

Introduction

In rostering problems, we need to find a schedule that satisfies various constraints. Ergonomic constraints, that place restrictions on a number of consecutive days, are common (Bourdais, Galinier, & Pesant 2003) . To model such rules, several global constraints have been introduced. For example, the STRETCH constraint (Hellsten, Pesant, & Beek 2004) is useful for limiting the length of stretches of variables (e.g., no more than 3 consecutive night shifts for an individual employee). The SEQUENCE constraint (Beldiceanu & Contejean 1994) is used to restrict the number of occurrences of specific values in a given period (e.g., each employee has to have 2 days-off in any 7 consecutive days). The REGULAR constraint (Pesant 2004) ensures that an assignment of a given sequence of variables belongs to a regular language. This can express, for example, that an employee has to have at least two consecutive days on any shift.

Many public services, like hospitals, police departments, some factories, work 7 days per week. Such organizations require a schedule than can be repeated with a given period. To express rules for such cyclic schedules the Cyclic REGULAR constraint (Quimper & Walsh 2006) and the Cyclic STRETCH constraint (Hellsten, Pesant, & Beek 2004) have been introduced. Propagating the Cyclic REGULAR constraint is NP-hard, whilst there is a polynomial filtering algorithm for the Cyclic STRETCH constraint. Recently, the Cyclic SEQUENCE has been introduced but no polynomial filtering algorithms have been proposed (Brand *et al.* 2007). In this paper we present a polynomial filtering algorithm for the Cyclic SEQUENCE. We also consider a special case of SEQUENCE, the ATMOSTSEQ constraint, and prove that the Cyclic ATMOSTSEQ constraint can be decomposed into ATMOST constraints without hindering propagation.

The SEQUENCE constraint

The AMONG constraint restricts the number of occurrences of some given values in a sequence of k variables. More precisely, $\text{AMONG}(l, u, [X_1, X_2, \ldots, X_k], v)$ holds iff $l \leq |\{i | X_i \in v\}| \leq u$. The AMONG constraint can be encoded by channeling into Boolean variables, namely, $X'_i = 1$ if $X_i \in v$, $X'_i = 0$ otherwise, and $l \leq \sum_{i=1}^{k} X'_i \leq u$ without hindering propagation. Consequently, we will simplify notation and consider AMONG (and SEQUENCE) on Boolean variables with $v = \{1\}$. If $l = 0$, AMONG becomes an ATMOST constraint. ATMOST is *monotone* since given a support, we also have support for any larger value (Bessiere *et al.* 2007). The SEQUENCE constraint is a conjunction of overlapping AMONG constraints. More precisely, $\text{SEQUENCE}(l, u, k, [X_1, \ldots, X_n])$ holds iff for $i = 1, \ldots, n-k+1$, $\text{AMONG}(l, u, [X_i, \ldots, X_{i+k-1}])$ holds. For instance, $\text{SEQUENCE}(1, 2, 4, [X_1, \ldots, X_n])$ ensures that in any sequence of four consecutive variables only one or two of variables can take the value 1.

We are very thankful to George Katsirelos for valuable discussions on this work.

The Cyclic SEQUENCE constraint

In scheduling problems, we might want a rotating schedule, which can be repeated in a cycle. To model this we can use a cyclic version of the SEQUENCE constraint. $\text{SEQUENCE}_{\odot}(l, u, k, [X_1, \ldots, X_n])$ ensures that between l and u variables in X_i to $X_{1+(i+k-2 \bmod n)}$ take value 1 for $i = 1, \ldots, n$.

We propose a polynomial filtering algorithm for the Cyclic SEQUENCE constraint. This algorithm is an extension of the $HPRS$ algorithm (Hoeve *et al.* 2006) for the SEQUENCE constraint. The core of the $HPRS$ algorithm is the `CheckConsistency` procedure that finds support for all domain values of all variables or proves that a support does not exist. In order to find a support for $v_j \in D(X_k) = \{0, 1\}$, an array of cumulative sums y of length $n + 1$ is introduced. A value y_i shows the number of ones in the first i variables X. The algorithm initially assigns values y as $y_i = \sum_{k=1}^{i} \min(D(X_k))$ for $i = 1$ to n and keeps repairing y's until they satisfy all constraints: $y_{i+1} - y_i \in D(X_{i+1})$, $i = 0, \ldots, n-1$ and $l \leq y_{i+k} - y_i \leq u$, $i = 0, \ldots, n-k$. It should be noted that there is a one-to-one correspondents between valid assignments X and solutions y for the SEQUENCE constraints. As was shown (Hoeve *et al.* 2006), `CheckConsistency` procedure returns the minimal solution $y^{\min}$, such that for any solution y of the SEQUENCE constraint holds that $y_i^{\min} \leq y_i$, $i = 1, \ldots, n$. Similarly, the algorithm can compute a solution $y^{\max}$, such that $y_i \leq y_i^{\max}$, $i = 1, \ldots, n$.

The core of the proposed algorithm for the cyclic SEQUENCE constraint consists of extension of the `CheckConsistency` procedure to the cyclic case, so

that, for all variable domain values if finds a *cyclic* support or proves that it does not exists. We will use the following running example throughout the rest of the paper to illustrate the algorithm. Consider $\text{SEQUENCE}_{\odot}(1, 2, 4, [X_1, \ldots, X_6])$, $D(X_i) = \{0, 1\}$,$i = 1, \ldots, 6$. Suppose we perform the consistency check for $X_4 = 0$. This is equivalent to finding a solution for cyclic SEQUENCE with $D(X_i) = \{0, 1\}$, $i \in [1, 2, 3] \cup [5, 6]$ and $D(X_4) = 0$.
The algorithm is based on two simple observations.

1. Consider the $\text{SEQUENCE}(1, 2, 4, [X_1, \ldots, X_9])$ constraint over extended sequence of variables. We introduce 3 extra variables X_7, X_8, X_9 to break cyclicity. Domains of extra variables are $D(X_{6+i}) = D(X_i)$, $i = 1, \ldots, 3$. The $\text{SEQUENCE}(1, 2, 4, [X_1, \ldots, X_9])$ constraint can be seen as relaxed version of $\text{SEQUENCE}_{\odot}(1, 2, 4, [X_1, \ldots, X_6])$, such that solutions of the cyclic SEQUENCE form a subset of its solutions. If a solution of $\text{SEQUENCE}(1, 2, 4, [X_1, \ldots, X_9])$ satisfies $X_{6+i} = X_i$, $i = 1, \ldots, 3$, then it is a solution of the cyclic SEQUENCE.

2. Consider a cumulative sum y_i. As mentioned above, it can take values in the interval $[y_i^{min}, y_i^{max}]$. Find the minimal solution y' of the SEQUENCE constraint such that y_i is fixed to w, $w \in [y_i^{min}, y_i^{max}]$ and the corresponding assignment for X's: $X_{j+1} = y'_{j+1} - y'_i$, $j = 0, \ldots, n-1$. Then X is the lexicographically smallest assignment among all valid assignments with y_i fixed to w. Moreover, the suffix of X, $X_{i+1}, \ldots, X_n$, is the lexicographically smallest suffix among suffixes of all valid assignments with y_i fixed to w. Coming back to the example, consider a cumulative sum y_6, which can take values from 1 to 4 (Figure 1a). If we fix y_6 to 1, the minimal solution of $\text{SEQUENCE}[X_1, \ldots, X_9]$ with $y_6 = 1$ is $y' = [0001111222]$. The corresponding assignment is $X = [001000100]$ which is lexicographically smallest among all valid assignments with $y_6 = 1$. The suffix of X, $[X_7, X_8, X_9] = [100]$ is the lexicographically smallest suffix among suffixes of all valid assignments with $y_6 = 1$.

Consider the minimal solution $y' = [0001111222]$ and the corresponding assignment $X = [001000100]$ of $\text{SEQUENCE}[X_1, \ldots, X_9]$, where y_6 is fixed to 1. From the second observation it follows that the prefix $[X_1, X_2, X_3] = [001]$ is the lexicographically smallest prefix among prefixes and the suffix $[X_7, X_8, X_9] = [100]$ is the lexicographically smallest suffix among suffixes of all valid assignments with $y_6 = 1$. Consider the earliest position where the prefix and the suffix are different. This is the first position: $X_1 = 0$ and $X_7 = 1$. From the first observation it follows that for a cyclic solution the prefix and the suffix of length 3 are equal. Then, there is no *cyclic* assignment for X with the suffix $[X_7, X_8, X_9] = [0, *, *]$, because the lexicographically smallest prefix is $[X_1, X_2, X_3] = [1, *, *]$. So, we can safely prune all assignments with prefixes with the value 0 at the first position.

This observation is the intuition behind the following Lemma 1.

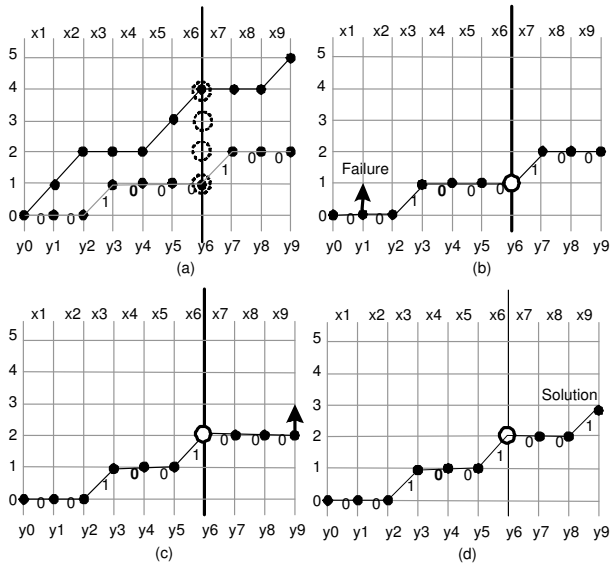

Figure 1: Finding a cyclic support of $\text{SEQUENCE}_{\odot}(1, 2, 4, [X_1, \ldots, X_6])$ for $X_4 = 0$

Lemma 1 *Consider the minimal solution y' of $\text{SEQUENCE}(l, u, k, [X_1, \ldots, X_{n+k-1}])$ with y_n fixed to w, $w \in [y_n^{min}, y_n^{max}]$. Let i be such position that $\forall h \in [0, i) : X_h = X_{n+h}$, and $X_i \neq X_{n+i}$. We assume that $X_i = 0$ and $X_{n+i} = 1$ (the case $X_i = 1$ and $X_{n+i} = 0$ is symmetric). Then y_i is greater than y'_i in any solution y of $\text{SEQUENCE}_{\odot}(X_1, \ldots, X_n)$ with $y_n = w$.*

Consider solutions of $\text{SEQUENCE}[X_1, \ldots, X_{n+k-1}]$ where y_n is fixed to w, $w \in [y_n^{min}, y_n^{max}]$. Lemma 1 allows us to increase the low bound of y_i for some $i \in [1..k-1] \cup [n+1..n+k-1]$ without losing any cyclic assignments. Then we re-compute the minimal solution y taking this new lower bound into account. These steps are repeated until we find a cyclic assignment or fail and select the next value of y_n. The pseudocode for the resulting cyclic consistency check procedure is presented as Algorithm 1. The algorithm uses an auxiliary procedure `CheckConsistencyEx`, which is a generalization of `CheckConsistency`. It finds the minimal solution for a constraint $\text{SEQUENCE} \wedge \mathcal{S}$, where $\mathcal{S}$ is a conjunction of primitive constraints of the form $\{y_i = p\}$ or $\{p < y_i\}$. The algorithm for `CheckConsistencyEx` is identical to the original `CheckConsistency` procedure except it adjusts the initial values of y's to satisfy $\mathcal{S}$ and check if a resulting solution y satisfy $\mathcal{S}$.

As can be seen from the Algorithm 1, we perform the main loop (lines 6–16) no more than $O(n)$ times. In a loop, we can do no more than $2(k-1)^2$ iterations before failing or finding a solution for $\text{SEQUENCE}(l, u, k, [X_1, \ldots, X_{n+k-1}])$ with y_n fixed to w. The cost of the `CheckConsistencyEx` procedure is $O(n^2)$. Consequently, the total time complexity of the `CheckConsistencyCyclic` procedure is $O((nk^2)n^2)$. As we perform it for all variables domain values, the total complexity of the filtering algorithm is $O((nk^2)n^3d)$.

Algorithm 1 Consistency check for the Cyclic SEQUENCE

```
1: procedure CHECKCONSISTENCYCYCLIC(X, D, k)
2:     for i ← 1 to k − 1 do
3:         D(X_{n+i}) ← D(X_i);
4:     if ¬CheckConsistency(y^min, y^max) then
5:         return 0;
6:     for w ← y_n^min to y_n^max do
7:         S ← {y_n = w};
8:         while (¬CheckConsistencyEx(S, y')) do
9:             for i ← 1 to k − 1 do
10:                x_i ← y'_i − y'_{i−1};
11:                x_{n+i} ← y'_{n+i} − y'_{n+i−1};
12:                if (x_i == 0) ∧ (x_{n+i} == 1) then
13:                    S ← S ∪ {y'_i < y_i}; break;
14:                if (x_i == 1) ∧ (x_{n+i} == 0) then
15:                    S ← S ∪ {y'_{n+i} < y_{n+i}}; break;
16:            return 1;
17:    return 0;
```

Consider how the algorithm works on the example.

1. Line 4: Find y_{min} and y_{max} solutions of SEQUENCE over extended sequence of variables $[X_1, \ldots, X_6, X_7, X_8, X_9]$ to get bounds for y_6, $y_6 \in [1, 4]$ (Figure 1(a)).
2. Line 6 (first iteration): Set y_6 to 1. Add the constraint $\{y_6 = 1\}$ to $\mathcal{S}$ and perform `CheckConsistencyEx` (Figure 1(b)), which returns the minimal solution $y' = [0001111\ 222]$; the corresponding assignment for X is $X = [001000\ 100]$. The first mismatch between the prefix and the suffix of length 3 is in the first position: $X_1 = 0$ and $X_7 = 1$. Hence, we add $\{y_1 > 0\}$ to $\mathcal{S}$ and invoke `CheckConsistencyEx`, which fails. So, we go to the next possible value of y_n.
3. Line 6 (second iteration): Set y_6 to 2. Add $\{y_6 = 2\}$ to $\mathcal{S}$ and perform `CheckConsistencyEx` (Figure 1(c)), which returns the minimal solution $y' = [0001112\ 222]$; corresponding assignment for X is $X = [001001\ 000]$. The first mismatch between the prefix and the suffix of length 3 is in the third position: $X_3 = 1$ and $X_9 = 0$. Hence, we add $\{y_9 > 2\}$ to $\mathcal{S}$ and invoke `CheckConsistencyEx`, which returns $y' = [0001112\ 223]$. This solution corresponds to a cyclic support $[001001]$ for $X_4 = 0$ (Figure 1(d)).

The Cyclic ATMOSTSEQ constraint

In this section we consider a cyclic version of the ATMOSTSEQ constraint. $\text{ATMOSTSEQ}_{\odot}(u, k, [X_1, X_2, \ldots, X_n])$ ensures that at most u variables in X_i to $X_{1+(i+k-2 \bmod n)}$ take value 1 for $i = 1$ to n. Decomposition of the ATMOSTSEQ constraint into set of ATMOST constraints does not hinder propagation because the ATMOST constraint is monotone (Bessiere *et al.* 2007). We show that decomposition of $\text{ATMOSTSEQ}_{\odot}$ into set of ATMOST constraints also does not hinder propagation. First, we prove a more general statement that domain consistency on the conjunction of any set of monotone constraints with a common order on domain values is achieved by enforcing domain consistency on the individual constraints.

Definition 1 *Let $\Gamma = \{C_1, \ldots, C_m\}$ be a set of monotone constraints. Γ is a* monotone *set of constraints iff there exists a* single *total ordering $\prec$ of the domain values such that for any two values v, w, if $v \prec w$ then w can be substituted for v in any solution for* any *C_i.*

Lemma 2 *If a set of constraints $\Gamma = \{C_1, \ldots, C_m\}$ is monotone then domain consistency on $\bigwedge_{i=1}^{m} C_i$ is equivalent to domain consistency on (C_i) for $i = 1, \ldots, m$.*

Proof: If $\bigwedge_{i=1}^{m} C_i$ is domain consistent then all C_i, $i = 1, \ldots, m$ are domain consistent. Suppose all C_i, $i = 1, \ldots, m$ are domain consistent. We show that $X_k = v_j$ has support for the conjunction of C_i for $i = 1$ to m. Consider the assignment where $X_k = v_j$ and all other variables are set to the first value in their domains from the total order $\prec$. This assignment satisfies all constraints because if a constraint C_i is not satisfied this way, then it does not have any support at all, hence it can not be domain consistent. Consequently, $X_k = v_j$ has a global support for $\bigwedge_{i=1}^{m} C_i$. $\diamond$

As a consequence of Lemma 2 and the fact that ATMOST is monotone, we get:

Corollary 1 *Enforcing domain consistency on $\text{ATMOSTSEQ}_{\odot}$ is equivalent to enforcing domain consistency on $\text{ATMOST}(u, [X_i, .., X_{1+(i+k-2) mod\ n)}])$ for $i = 1$ to n.*

References

Beldiceanu, N., and Contejean, E. 1994. Introducing global constraints in CHIP. *Mathematical and Computer Modelling* 12:97–123.

Bessiere, C.; Hebrard, E.; Hnich, B.; Kiziltan, Z.; and Walsh, T. 2007. The SLIDE-meta constraint. TR.

Bourdais, S.; Galinier, P.; and Pesant, G. 2003. Hibiscus: A constraint programming application to staff scheduling in health care. In *Proc. of the 9th Int. Conf. on Principles and Practice of Constraint Programming*, 153–167.

Brand, S.; Narodytska, N.; Quimper, C.-G.; Stuckey, P.; and Walsh, T. 2007. Encodings of the sequence constraint. TR COMIC-2007-010.

Hellsten, L.; Pesant, G.; and Beek, P. v. 2004. A domain consistency algorithm for the stretch constraint. In Wallace, M., ed., *In Proc. of the 10th Int. Conf. on Principles and Practice of Constraint Programming*, 290–304.

Hoeve, W.-J. v.; Pesant, G.; Rousseau, L.-M.; and Sabharwal, A. 2006. Revisiting the Sequence Constraint. In Benhamou, F., ed., *Proc.of the 12th Int. Conf. on Principles and Practice of Constraint Programming*, 620–634.

Pesant, G. 2004. A regular language membership constraint for finite sequences of variables. In Wallace, M., ed., *Proc. of 10th Int. Conf. on Principles and Practice of Constraint Programming*, 482–495.

Quimper, C.-G., and Walsh, T. 2006. Global grammar constraints. TR COMIC-2006-005.