ウェブコンテンツ配信スケジューラのための制約処理系の設計

Design of A Constraint Solver for distributing Web Contents

平石 広典 † 溝口 文雄 † ††
Hironori HIRAISHI Fumio MIZOGUCHI
† 株式会社ウィズダムテック †† 東京理科大学理工学部
WisdomTex Inc. Faculty of Sci. and Tech., Tokyo Univ. of Science
hiraishi@wisdomtex.com

本研究では，広告や音楽，動画などのウェブコンテンツにおけるスケジューラのための制約処理系を設計した．様々なコンテンツがウェブ上に存在し，それらはユーザの好みで選択できるものであるが，それらの順番をスケジューリングすることで，ユーザごとの番組として提供することができる．その場合，全体の時間，ユーザの好み，コンテンツの順番，広告の割り込み，などの制約が存在する．本研究ではこのような課題に対して制約処理によってアプローチし，そのための制約処理系の実装を行った．

1 はじめに

音楽データをはじめとして，スポーツやニュースなどの様々な動画コンテンツをウェブ上から入手可能となってきた．実際に，YAHOO や GYAO[1]などを利用して動画コンテンツを無料で閲覧することが可能であり，各プロバイダサイトにおいても，さかんに動画コンテンツの配信が行われている．

今現在，それらのコンテンツは個々のコンテンツ毎にユーザの選択によって，ダウンロード，もしくは閲覧されるのが一般的である．しかしながら，それらのコンテンツの順番をスケジューリングすることで，ユーザごとの番組として提供することができる．その場合，全体の時間，ユーザの好み，コンテンツの順番，広告の割り込みなどの制約が存在する．

本研究ではこのような課題に対して制約処理 [3, 4, 5] によってアプローチし，そのための制約処理系 SYK (SeiYaKu) の設計・実装を行った．ウェブコンテン配信スケジュールを階層的に表現し，スケジューリングに必要な制約の定義を行った．そして，定義された問題を解くための制約処理系を Java 言語のライブラリとして実装を行った．

以下，第 2 章では，ウェブコンテンツ配信スケジュールの表現を定義し，第 3 章では，定義されたスケジュールを解くための制約処理系の詳細について述べる．第 4 章でまとめと今後の課題について述べる．

[1] http://www.gyao.jp

2 ウェブコンテンツ配信スケジュール

動画，音楽，広告などを含むウェブコンテンツのスケジューリングによって生成される解（スケジュール）は，コンテンツの並びである．コンテンツには，カテゴリやジャンルなどの属性データが付随するのが一般的である．そのため，本研究で生成するスケジュールは属性ごとに階層構造で表現する．

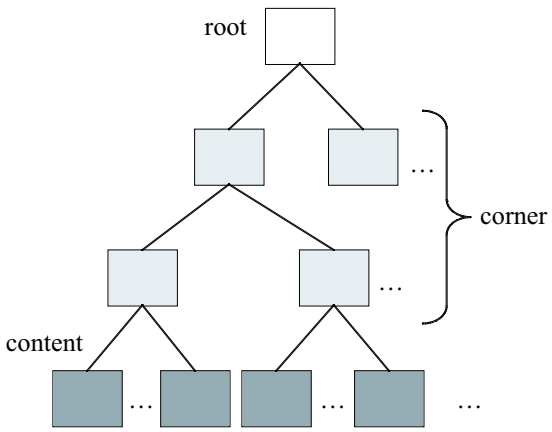

図 1: スケジュールの階層構造

図 1 にはスケジュールの階層構造を示した．最上位の root はスケジュール全体を意味しており，最下位の content はコンテンツそのものを意味する．root と content の間の corner 層であり，コンテンツに付随する属性を意味する．つまり，コンテンツの属性

に応じて，コーナーを設け，例えば，ニュースコーナー，スポーツコーナーなどをまとめるかたちでスケジューリングを実行する．corner 層の階層数は任意である．本研究におけるウェブコンテンツ配信スケジューラは，このように階層的に表現された content や corner に対して，各々のコンテンツを割り当てるといったタスクを実行する．

2.1 エレメントオブジェクト

スケジュールは階層構造によって表現したが，その要素となる root, corner や content は統一的に表現することができる．以下はそのためのエレメントオブジェクトに与える変数を示した．

index :エレメントに割り当てたコンテンツの ID.

duration :割り当てたコンテンツの時間間隔.

startsAt :コンテンツの開始時刻.

endsAt :コンテンツの終了時刻.

parent : 1 つ上位のエレメント．null の場合には root を意味する.

children :下位のエレメントの順列．null の場合には content を意味する.

localPrev :同じ上位のエレメント（コーナー）内での次（右隣）のエレメント．null の場合はそのコーナー内の最初のエレメントを意味する.

localNext :同じ上位のエレメント（コーナー）内での前（左隣）のエレメント．null の場合はそのコーナー内の最後のエレメントを意味する.

globalPrev :同じ階層の次（右隣）のエレメント．null の場合はその階層の最初エレメントを意味する.

globalNext :同じ階層の前（左隣）のエレメント．null の場合はその階層の最後エレメントを意味する.

上記の変数を与えることで，図 1 に示した階層構造が表現される．そして，スケジューラはエレメントの *index* を具体的なコンテンツの ID に割り当てることで，実際のスケジュールが生成されることになる.

2.2 0 時間コンテンツの利用

スケジューラは生成されるスケジュールに，予めいくつのコーナーやコンテンツが利用されるかを，予め知ることはできない．そのため，最初に図 1 のような階層構造を用意する際には，実際に利用されるコーナーやコンテンツ数よりも多くのエレメントを用意しておく必要がある[2]．しかしながら，全体の時間などの制約を満たしていれば，用意した全てのエレメントに具体的なコンテンツを割り当てる必要はない．例えば，コーナーに対してコンテンツを割り当てるためのエレメントを 10 個用意し，そのコーナーの時間を 60 分とした場合，仮に 20 分のコンテンツを 3 つ割り当てれば，残りの 7 個のエレメントにコンテンツを割り当てる必要はない.

そこで本研究では，時間間隔が 0 時間のコンテンツを仮想的に用意し，割り当てる必要のないエレメントに対して，0 時間コンテンツを割り当てることにした．これによって，コーナーやコンテンツ数を可変にすることができ，スケジューラの処理をエレメントに対してコンテンツを割り当てるといった単一にすることが可能である.

2.3 制約の定義

スケジュールの生成に利用される制約を表 1 に示した.

表 1: スケジュールに利用される制約

	メソッド	引数	意味
時間間隔 ($duration$)	setDuration	T	$duration = T$
	setMaxDuration	T	$duration \leq T$
	setMinDuration	T	$duration \geq T$
開始時刻 ($startsAt$)	startsAt	t	$startsAt = t$
	startsMaxAt	t	$startsAt \leq t$
	startsMinAt	t	$startsAt \geq t$
終了時刻 ($endsAt$)	endsAt	t	$endsAt = t$
	endsMaxAt	t	$endsAt \leq t$
	endsMinAt	t	$endsAt \geq t$
回数 ($count$)	setCount	n	$count = n$
	setMaxCount	n	$count \leq n$
	setMinCount	n	$count \geq n$
周期 ($period$)	setPeriodic	T	$period = T$
	setMaxPeriodic	T	$period \leq T$
	setMinPeriodic	T	$period \geq T$
順序	startsAtEnd	C	C の直後に開始
	startsAfterEnd	C	C の後に開始
	endsAtStart	C	C の直前に終了
	endsAtStart	C	C の前に終了

時間間隔は全体のスケジュールや各々のコーナーやコンテンツに与える制約で，スケジュール全体の長さ

[2]最悪の場合，全てのコンテンツ数分のエレメントを用意する必要がある.

や各コーナーの長さを制約する．開始時刻，終了時刻は，特定の時刻にコンテンツやコーナーを開始するための制約である．回数は同一のコーナーやコンテンツを何回利用するかを制約するものである．本研究のスケジューラでは，デフォルトでは各々のコーナー，コンテンツはスケジュールで 1 回以下 (setMaxCount(1)) としている．ここで，setCount(n) は，必ず n 回利用されることを意味し，setMaxCount(n) は n 回以下であり利用されなくてもいい．setMinCount(n) は n 回以上を意味しており，この場合には，必ず n 回は利用されなければならない．

周期はある一定の時間ごとに利用されなければならないコーナー，コンテンツに対して与えられる制約で，広告などのに利用される．最後に順序は二つのコーナー，コンテンツ間の順序関係を制約するものである．A.startsAtEnd(B) の場合，A を開始する場合には必ず B の直後に開始しなければならないことを意味しており，B の直後に必ず A がくる必要はない．また，A.startsAfterEnd(B) では，A を開始するのは B が終わった後（直後である必要はない）でなければならないことを意味しており，同様に B の後に必ず A がくる必要はない．一方，B.endsAtStart(A) は，B は A が開始する直前に終了することを意味していおり，この場合，B の直後に必ず A が開始しなければならない．また，B.endsAfterStart(A) は，B は A が開始する前（直前である必要はない）に終了することを意味しており，同様に，B の後に必ず A が開始しなければならない．これは，切れ目のないノンストップの音楽を流す場合や時系列的に関係のあるコンテンツを選択する場合に利用される．

デフォルトの制約として，各エレメントでは

$$startsAt{+}duration{=}endsAt$$

といった制約が与えられる．また，上位のエレメントオブジェクトの時間間隔は，下位のエレメントの時間間隔の総和となるため，

$$duration_{upper}{=}\sum_{i=0}^{n} duration_{lower-i}$$
（n は下位のエレメント数）

といった制約が与えられる．さらに，上位のエレメントの開始時刻とそのエレメントに含まれる下位のエレメントの最初のエレメントの開始時刻は同じであり，上位のエレメントの終了時刻とそのエレメントに含まれる下位のエレメントの最後のエレメントの終了時刻は同じであるため，$startsAt_{upper}{=}startsAt_{lower-0}$，および，$endsAt_{upper}{=}endsAt_{lower-n}$ といった制約が与えられる．

3 制約処理系 SYK

前章で説明したスケジュールを生成するための制約処理系 SYK(SeiYaKu) を設計・実装した．

3.1 制約のオブジェクト構造

図 2 には SYK の制約のオブジェクト構造を示した．

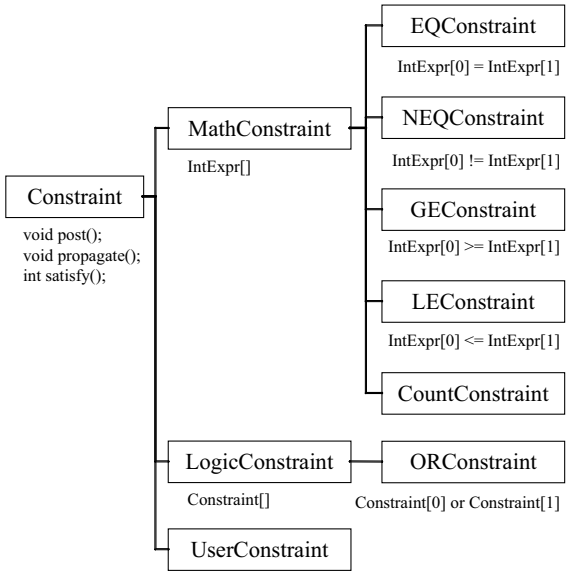

図 2: 制約のオブジェクト構造

SYK で扱う制約は全て Constraint オブジェクトを継承する．Constraint オブジェクトにはこの制約が定義された時に最初に呼び出される post メソッド，この制約に利用されている変数が具体化された時に呼び出される propagate メソッド，それと，この制約を満たしているかどうかを判断する satisfy メソッドの 3 つのメソッドが定義されている．

SYK では 3 つの制約を用意した．MathConstraint オブジェクトは，数値 (IntExpr オブジェクトで表現) を扱う算術的制約を処理するためのもので，それを継承した EQConstraint，NEQConstraint，GEConstraint，LEConstriant，CountConstraint オブジェクトを定義した．

LogicConstraint は，制約 (Constraint オブジェクトで表現）どうしの論理的制約を扱うもので ORConstraint オブジェクトを定義した．AND に関しては特に定義する必要はなく，与えられた制約は全て満たす必要があるため，各々の制約は基本的に全て AND

の関係にある．UserConstraint は，ユーザが自由に定義できる制約である．

3.2 制約処理の流れ

以下は SYK の制約解消処理の基本的な流れである．

```
1.  private boolean next(){
2.    IntVar iv = null;
3.    curentvar = 0;
4.    while(currentvar < solvevars.size()){
5.      if(currentvar < 0){
6.        return false;
7.      }
8.
9.      iv=(IntVar)
           (solvevars.elementAt(currentvar));
10.
11.     if(!iv.addPoint()){
12.       currentvar = currentvar-1;
13.       iv.clear();
14.     }
15.     else{
16.       currentvar++;
17.     }
18.   }
19.   return true;
20. }
```

4 行目の solvevars は値を割り当てる変数の順列を示すグローバル変数である．また，currentvar は現在何番目の変数に対して値の割当を行っているかを示すインデックスを示す．5 行目で currentvar<0 の時は，最初の変数の割当に失敗したことを意味し，解なしとして false を返して終了する．9 行目で実際に割当を行う変数 iv を取り出している．11 行目の iv.addPoint() によって，iv に対しての値の割当を実行する．12 行目は割当に失敗した場合の処理で，現在のインデックスを 1 つ前の変数に戻し，13 行目で割当に失敗した変数の初期化を行っている．iv.clear() では，iv を割当処理を行った際に制約伝搬によって割り当てられた他の変数の値をもとに戻したり，新たに追加された制約を削除したりしている．割当が成功した場合には，16 行目で変数のインデックスを次の変数のものにしている．

3.3 総和の制約解消

ウェブコンテンツ配信スケジュールにおいて基本的な制約は，第 2.3 節で述べたように時間間隔である．したがって，時間間隔の制約をうまく処理することで，全体のスケジューリングにかかる時間を短縮することができる．しかしながら，実際に扱う時間間隔は下位のエレメントの時間間隔の総和として表現される．そのため，時間間隔の制約が与えられたとしても，総和を構成する組み合わせは多く，1 つの変数が具体化したといって他の変数をフォワードチェックによって変数の候補を削減するのは難しい．しかしながら，変数を削除できない場合，早い段階で失敗の状態を発見できれば，早い段階で次の候補を試すことが可能で，より速く解を発見することができる．

ここで比較する変数 $V1$,$V2$ を考えると，$V1$ の最小値を $min1$，最大値を $max1$ とし，$V2$ の最小値を $min2$，最大値を $max2$ とする．その場合，SYK では変数の割当を行って以下の関係となってしまった場合には失敗と判断し，バックトラックを行うようにしている．

$$max2 < min1 \ or \ max1 < min2$$

これは，二つの変数に重なりがないことを意味している．ここで，実際には各々の変数は総和となるため，最小値は構成する各変数の最小値の和となり，最大値は構成する各変数の最大値の和となる．

4 まとめ

本稿では，ウェブコンテンツ配信スケジュールをコンテンツの属性ごとの階層構造として表現した．また，そのスケジュールを生成するための制約処理系 SYK の設計と実装を行った．解が存在しない場合には制約階層 [1] による制約の除去を行う．また，コンテンツそのものに評価値をもうけており，再スケジューリング [2] の際には，以前のスケジュールの評価値を上げることで，前回と違いの少ないスケジュールを生成することが可能である．

参考文献

[1] Alan Borning, Bjorn Freeman-Benson, and Molly Wilson, Constraint Hierarchies, *Lisp and Symbolic Computation*, Vol. 5 No. 3, pp.223-270, 1992.

[2] G. Verfaillia and T. Schex, Solution reuse in dynamic constraint satisfaction problems, *Proc. of the 12th National Conference on Artificial Intelligence*, pp.307-312, 1994.

[3] Krzysztof R. Apt, *Principles of Constraint Programming*, Cambridge University Press, 2003

[4] 溝口文雄，ジャン・ルイ ラッセ，古川 康一，制約論理プログラミング，共立出版, 1989

[5] Rina Dechter, *Constraint Processing*, Morgan Kaufmann Publishers, 2003