人　工　知　能　101－5
（1995．9．12）

制約違反最少化戦略による対話型時間割編成システム

西森 雄一　　内野 寛治　　狩野 均　　西原 清一

筑波大学　電子・情報工学系

大学の時間割編成問題を制約充足問題として捉え、制約により問題を表現し、解を得る対話型システムを開発した。時間割編成問題は探索空間が膨大で、多種多様なユーザの要求をすべて制約で表現することは難しいという問題点があり、自動的に実用的な時間割を得ることは困難である。そこで、本システムでは探索をユーザの指示によって行うものとして、システムとユーザが協調して探索を進めることにした。また探索手法として、制約違反点数を導入した制約違反最少化戦略による山登り法を提案した。本システムの特徴は、ユーザに制約違反の情報を示しながら対話によって時間割を編成できるところにある。

INTERACTIVE TIMETABLE SCHEDULING SYSTEM USING MIN-CONFLICTS HILL-CLIMBING METHOD

Yuuichi Nishimori　Kanji Uchino　Hitoshi Kanoh　Seiichi Nishihara

Institute of Information Sciences and Electronics
University of Tsukuba
Tsukuba, Ibaraki　305,　Japan

We developed an interactive system to solve the university timetabling problems which are considered as constraint satisfaction problems. The timetabling problems have the following problems: First, search space is very large. Second, it is difficult to represent all various kinds of requirements by constraints. Therefore it is hard to produce practical timetables automatically. In order to resolve it, our approach takes advantage of interactions in cooperating with the user to guide its search. We propose the min-conflicts hill-climbing method with violation points for timetabling problems. Our system can make timetables interactively by showing information of constraint violations.

1 はじめに

時間割編成問題とは、各科目が実施される時間帯をまとめた時間割を編成する問題のことである。時間割編成は、制約充足問題 [1] として多くの研究がなされている [2, 3, 5]。しかし、時間割編成問題を解くにはカリキュラムに関する専門的な知識が必要で、その多くが計算機内で表現しにくく、表現できても膨大な計算時間と記憶領域を要する。また、それぞれの事例ごとに制約条件が異なるために有効な解法が見つけられておらず、編成は人手に頼らざるを得なかった。以上のような背景から、実際の時間割編成に対応できる問題の表現方法や、それを効率的に解く手法が望まれている。

現在開発されている時間割編成システムには、以下のものがある。吉田ら [3] のシステムは、制約の強弱に注目して、最適化の手法を導入した拡張制約表現によって構成されている。また Cooper ら [2] のシステムは、実際の高校の時間割編成の要求事項を簡単な言語で表現し、ヒューリスティクスを組み込んだ木探索を行って編成を行うものである。これら自動編成のシステムは、編成のための制約条件が限られており、一般に大学などの複雑な時間割をうまく編成することは困難である。また、編成中に人間の知識を効率的に活用することができないという問題点がある。

本システムでは、より現実的な時間割の編成を目的とする。上記のような従来システムが、自動的に探索して解を得ているのに対して、本システムではユーザと対話することで編成を行うものである。つまり、本システムの特徴は、対話によりシステムとユーザが協調して割当や探索を行うことにある。これにより、時間割編成にユーザの多種多様な要求を反映させることができる。

本稿では、まず時間割編成問題を筑波大学第三学群情報学類の実例より定式化し、時間割編成における課題や問題点を整理する。さらに、その対策として、本システムでとった方針を示す。次に、提案する手法について、制約の表現および探索アルゴリズムなどを詳しく述べる。最後に、システムの処理手順、実行例を示す。

2 問題点と対策

2.1 時間割編成問題

時間割編成問題とは、基本的に科目を実施する学年、および時間帯を決定することである。但し、「科目には担当する教官がおり、その教官が担当可能なように時間を決定すること」などの幾つかの制約が存在しており、それらを満たすように時間割を編成しなければならない。

本稿では実例として、筑波大学第三学群情報学類の時間割編成問題について取り上げる。この問題を次の 4 つ組で定義する。

$$\text{時間割編成問題} = (U, L, C, W)$$

$U = \{\text{科目}1, \ldots, \text{科目}M\}$: 科目 i の集合

科目 i には以下の属性が定義されている (表 1)。

- $name(i)$: 科目名
- $teacher(i)$: 担当教官
- $grade(i)$: 履修学年
- $course(i)$: 専攻グループ
- $group(i)$: 分野グループ

科目 i の次の属性は、未定義である。

- $slot(i)$: 割当先時間帯

$L = \{l_1, \ldots, l_N\}$: 時間帯 l_j の集合

$l_j = (\text{学期}, \text{曜日}, \text{時限})$

$C = \{c_1, \ldots, c_P\}$: 制約 c_k の集合

$W = \{w_1, \ldots, w_P\}$: 制約 c_k の重み w_k の集合

時間割編成問題を解くということは、全ての科目をある時間帯に、種々の制約を満たすように割り当てることである。ここで、科目 i をある時間帯 l_j に割り当てるとは、当初、未定義である $slot(i)$ に対して、$slot(i) = l_j$ を新たに追加することである。

制約 c_k の内容は表 2の通りである。表 2は、本システムで実現した制約である。表中の分類、違反点数は後述する。

2.2 時間割編成における問題点

時間割編成システムにおける課題は、次の 2 点である。

(1) 制約の表現方法の開発

(2) 探索アルゴリズムの開発

表 1: 科目データの一部 (筑波大学情報学類)

No	name	teacher	grade	course	group
1	代数学	池辺	1	必修	数学
2	代数学演習		1	必修	数学
3	幾何学	伊藤	1	必修	数学
30	情報科学概論 2A	井田	2	必修	情報科学
31	情報科学概論 2B	五十嵐	2	必修	情報科学
32	情報科学概論 2C	中田	2	必修	情報科学
33	非数値処理概論	西原, 田中	2	通常 2 年	情報処理
102	数理計画法 1	稲垣, 久野	3	3 年共通	情報数学
110	情報理論 1	海老原	3	情報科学	理論
130	最適化工学	寅市, 大塚	3	情報工学	情報数学
	⋮				

表 2: 時間割編成における制約

分類	C	時間割編成に含まれる制約	違反点数
科目単位	c_1	科目は割当可能な時間帯に割り当てられる。	10
	c_2	教官の都合のよい時間帯をとる。	10
	c_3	教官の意向を考慮した時間割を実現する。	5
2科目間	c_4	2科目間に関係が記述できる。	10
	c_5	2科目が学期集中のとき、なるべく時限連続にする。	2
	c_6	5が満たされないときは異なる曜日に割り当てる。	3
	c_7	2科目が学期連続のとき、なるべく同曜日同時限に割り当てる。	2
	c_8	2科目が時限連続のとき、なるべく昼休みを挟まない。	3
全体	c_9	教官は同時に2つ以上の時間帯を担当できない。	10
	c_{10}	科目には *course* の指定があり、裏番組は異なる *course* の科目同士でのみ作ることができる。	10
	c_{11}	3つ以上の科目を裏番組として同じ時間帯に割り当てることはできない。	10
	c_{12}	裏番組はなるべく作らない。	1
	c_{13}	低学年および高学年の必修科目を同時に実施しない。	3
	c_{14}	同曜日に実施される同分野の科目の時間数を制限する。	3

(1) の問題点としては、制約が多種多様であり、時間割編成システムに表現しにくく、または表現しきれないことである。

表 2に挙げた 14 個の制約のように、一般に満たされなければならない制約 (表 2では違反点数が 10 点) と、満たされることが望ましい制約 (同 1 ～ 5 点) がある。すなわち、これらの制約には重要度が存在する。また、個々の科目に対する科目単位制約、特定の 2 科目間に対する 2 科目間制約、時間割表の状態に関する全体制約がある。このように時間割編成問題の制約は多種多様である。

さらに時間割編成システムでは、複雑なカリキュラムを全て制約で表現しきれないことが多い。例えば、科目「情報工学基礎実験」では、実際には計算機台数の制限により学生を複数のグループに分け、講義と演習を交互に実施しているが、これを制約条件として実現することは困難である。また、一般に時間割編成問題では、多数の解の存在が予想され、この中から自動的に探索して、ユーザの要求を満たす解を得ることは困難である。

(2) の問題点としては、膨大な探索空間が挙げられる。

時間割編成問題は問題の規模が大きく、制約の種類も多いため、探索空間が膨大である。例えば科目数 $|U|$ を 100, 時間割表の時間帯数 $|L|$ を 75 とすると、探索空間のサイズは 10^{187} である。このため、通常の木探索法などでは対処しきれない。

2.3 基本方針

2.2 節の (1) の対策として、本システムでは次の 2 つの方針をとることにした。

① 重要度の高い制約を違反したときに、より多くの罰則を与えられるように、制約の重要度を違反点数で表現する

② システムで表現が困難な制約を時間割編成に活用させるために、ユーザと協調して探索を進める対話型システムとする

制約には重要度があるために、一般に制約の取り扱いが繁雑になるが、①のように違反点数を導入することで、違反点数の合計点数が最少になる方向に探索を進めるというように、取り扱うことができる。

②の実現のために、本システムは次の 2 つの編成モードを用意し、ユーザが適宜切り替えて、時間割を編成できるようにした。

- 手動編成　ユーザが科目表から科目を選択して時間割表に割り当てる。または割当解除する。
- 半自動編成　ユーザと協調して探索を進める。

手動編成は、表現された制約による機械的な割当だけでなく、表現が困難な制約、または人間が簡単に判断できることを時間割に直接反映させるために用意したものである。半自動編成は、システムが時にはランダムな選択によって自動的に探索を進めるのではなく、ユーザが介入しやすいように情報を与え、ユーザが選択をして探索を進めるものである。

さらに、2.2 節の (2) の対策として、

③ 制約違反点数を最少化する戦略を用いた山登り法を導入する

ことにした。

本手法の探索アルゴリズムには、本問題のように膨大な探索空間でも効率的に働くことが求められるため、確率的探索アルゴリズムを採用することにした。

確率的探索アルゴリズムには、山登り法 (HC)、焼きなまし法 (SA)、遺伝的アルゴリズム (GA) などがあるが、

- ②による対話処理の導入のため、アルゴリズムがシンプルである方がよい
- 制約違反点数を最少化する戦略を簡単に組み込めること

などにより、山登り法を選んだ。

また、山登り法では局所最適に陥ることがあり、その解決策が問題となる。この解決策としては、別の初期値からやり直す方法 (Iterated HC[6])、その局所最適の付近に脱出する方法 (Breakout Method [7]) などがあるが、本システムは対話型システムとすることから、対話操作でユーザが指示して局所最適から脱出することにした。

3 時間割編成システム

3.1 制約の表現方法

本システムでは、制約は手続きで表現されている。本システムで導入した制約は 2.2 節の表 2で示した 14 個であり、それらは次のように大まかに 3 つに分類できる。

$$C = \{ C_U , C_R , C_E \}$$

これらの制約の定義を以下に示す。

(1) 科目単位制約 C_U

科目単位制約は、制約の対象が科目単位である制約のことである。科目単位制約には以下の制約がある。

$$C_U = \{ c_1 , c_2 , c_3 \}$$

科目単位制約は、次の表 3-a の形式の表を参照する (c_1 は表 3-a、c_2, c_3 は付録の表 3-b,3-c)。表の左欄は、科目番号または教官名である。表の右欄には、(学期, 曜日, 時限) で表記された時間帯が入る。時間帯の 3 つの要素には、1 つの値もしくは、[値, 値,...] の形式である値のリスト、全ての値を示す * が入る (例えば、科目番号 29:「新入生オリエンテーション」は、1 学期実施のため、2,3 学期の全ての時間帯は割当不可能である)。

科目単位制約 c_1 の制約充足判定の内容は次の通りである (c_2, c_3 についても同様。付録参照)。

c_1 : 科目 i の割当時間帯 $slot(i)$ が表 3-a に入っていれば、制約違反。

表 3-a: 割当不可能時間帯表

科目	割当不可能時間帯
29	([2,3], *, *)
56	(*, *, [1,2])
⋮	⋮

表 4: 2 科目間関係表

(科目 k_1 , 科目 k_2)	関係
(1 , 2)	時限連続
(1 , 3)	学期集中
(30 , 31)	学期連続
(31 , 32)	学期連続
⋮	⋮

(2) 2 科目間制約 C_R

2 科目間制約は、2 つの科目間の制約条件のことである。

2 科目間関係は次の表 4のように与えられている。

表 4の関係の定義は以下の通りである。

- 時限連続 … 科目 k_1 と科目 k_2 の割当時間帯の時限が連続である。
- 学期連続 … 科目 k_1 と科目 k_2 の割当時間帯の学期が連続である。
- 学期集中 … 科目 k_1 と科目 k_2 の割当時間帯の学期が等しい。

2 科目間制約には次のものがある。

$$C_R = \{ c_4 , c_5 , c_6 , c_7 , c_8 \}$$

2 科目間制約 c_4, c_5 のそれぞれの制約充足判定の内容は次の通りである (c_6, c_7, c_8 については付録に示した)。

c_4 : 科目 k_1, k_2 の割当が、関係を満足していなければ、制約違反。

c_5 : 科目 k_1, k_2 の関係が学期集中であるとき、これらの関係が時限連続でないならば、制約違反。

(3) 全体制約 C_E

これは、時間割表全体についての制約である。全体制約の手続きは、制約によって異なる。

$$C_E = \{ c_9 , c_{10} , c_{11} , c_{12} , c_{13} , c_{14} \}$$

全体制約 c_9, c_{11} のそれぞれの制約充足判定の内容は次の通りである ($c_{10}, c_{21}, c_{13}, c_{14}$ については付録に示した)。

c_9 : 任意の科目 $i_1, i_2 (i_1 \neq i_2)$ に対して、$teacher(i_1) = teacher(i_2)$、かつ、$slot(i_1) = slot(i_2)$ のとき制約違反。

c_{11} : 全ての科目 i $(i = 1, \ldots, M)$ に対して、$slot(i)$ が等しいものの個数が3個以上ならば制約違反。

3.2 制約違反点数

2.2節で示した制約の重要度を実現するために、それぞれの制約に制約違反点数を導入する。

ある科目割当が制約に違反しているとき、その制約の違反点数をその科目に加算する。これを次のように定義する。

$$conf(i, c_k) = \begin{cases} w_k \cdots & \text{科目 } i \text{ の割当が} \\ & \text{制約 } c_k \text{ に違反} \\ 0 \quad \cdots & \text{その他} \end{cases}$$

科目 i の違反点数:

科目 i の違反点数は、その科目が違反している全ての制約の違反点数の合計である。

$$W_i = \sum_{k=1}^{P} conf(i, c_k)$$

総合違反点数:

総合違反点数は、編成中の時間割の制約違反の程度を表す。この総合違反点数を最少化することが、本システムの目的である。

$$W_{total} = \sum_{i=1}^{M} W_i$$

3.3 制約違反点数最少化戦略

本システムの半自動編成では、探索手法として制約違反最少化戦略による山登り法 [4] を次のように適用した。

制約違反点数最少化戦略:

評価値を総合違反点数の逆数として、総合違反点数が少なくなる方向に探索を進める。

制約違反最少化戦略は次の2ステップを行う [4]。

Step1 制約違反をしている変数を1つ選択する。
Step2 選択された変数に、制約違反が最少になるような値に変更する。

これを本手法では、次のように対応させた。

Step1 $\cdots$

W_i が最大となる科目 i $(i = 1, \ldots, M)$ を R とする。複数ある場合はユーザが選択する。

Step2 $\cdots$

時間割編成問題では、割当の変更は次の3通りが考えられる。変更方法として、この3つの方法から選択し、最少違反点数になるように変更することで、制約違反を改善する。

(a) 移動

科目 R を移動させることで、W_{total} が最少となるような時間帯に科目 R を移動する。

移動は、時間割表の科目を他の時間帯に移動させることである (図1参照)。

(b) 別科目割当

科目 R を割当解除し、その時間帯に W_{total} が最少となるような科目を割り当てる。

割当解除とは、時間割表から科目をはずし、科目表に戻すことである。すなわち、これは科目 R の割当時間帯 $slot(R)$ を未定義に戻す (図2参照)。

(c) 交換

科目 R の割当との交換により、W_{total} が最少となるような割当と交換する。

交換は、時間割表にある科目どうしで行う (図3参照)。

図1,2,3では、太枠で囲まれた解析学が科目 R であり、各候補の右側にその候補に変更したときの総合違反点数が表示されている。図1では候補は時間帯、図2,3では科目である。ここで、総合違反点数が最少となる時間帯または科目の候補に丸印が付けられており、この候補に対して上記 (a) ~ (c) の操作を行うことを示している。また、各候補は改善の程度によって、10点以上改善のとき白、1点以上10点未満改善のとき青、改善されないとき赤で色分け表示されている。

3.4 知識の導入

前節では、科目 R の割当を変更して、制約違反を減少させる手法を示した。しかしながら、時間割編成問題では実際に科目間に関係があり、それらの関係を考慮しなければ、制約違反を効率的に減少させることができない。

例)

科目 k_1 と科目 k_2 が時限連続のとき、科目 k_1 のみを変更しても (例えば他の時間帯に移動しても)、連続性の関係を満たさなくなり、結果として改善されない (図4)。

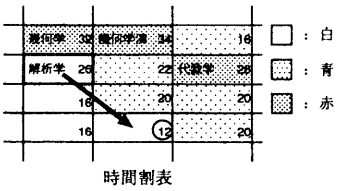

図 1: 変更方法が移動の場合

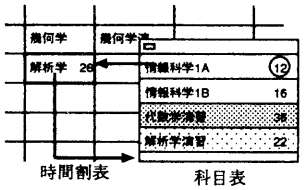

図 2: 変更方法が別科目割当の場合

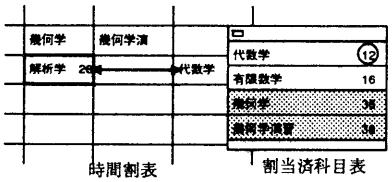

図 3: 変更方法が交換の場合

そこで本手法において、より効率的な探索を行うために、次の知識を導入する。

関連科目の一括改善:

2科目間関係によって関係付けられている科目を一括して改善する。

この知識は、時間割編成において2科目間の関係で、特に連続性を考慮したものである。

3.5 処理手順

本システムの処理手順を図 5-a,5-b,5-c に PAD で示す。

(a) 手動編成

手動編成は、ユーザはまず科目表から科目を選択し、システムからの以下の情報を参照しながら、時間割表のコマに割り当てを行うものである。

・時間割表のコマを違反点数によって色分けして表示。

・マウスカーソルで指されているコマについて違反している制約の内容を表示。

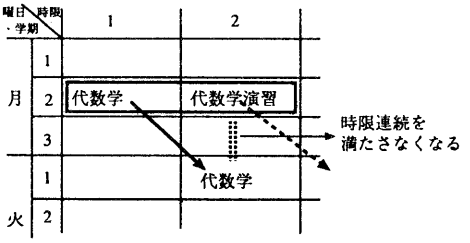

図 4: 関連科目の一括改善の例

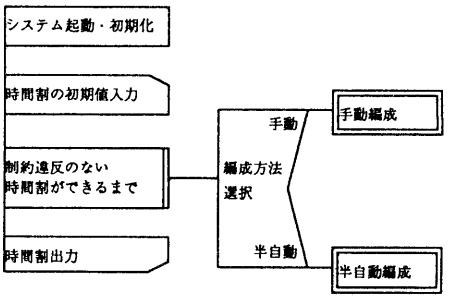

図 5 -a システム全体の処理

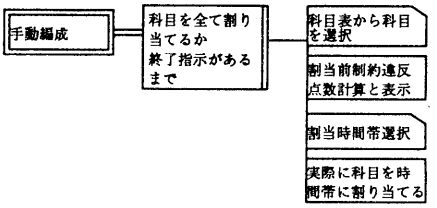

図 5 -b 手動編成の処理

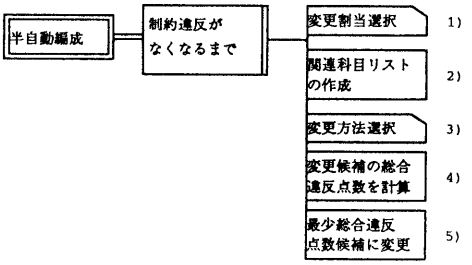

図 5 -c 半自動編成の処理

(b) 半自動編成

半自動編成では、次の5ステップ（図5-cの1)～5)に対応）で探索を行う。

1) システムは、現在編成中の時間割の制約違反を減少させる改善策として、まず時間割表から制約違反の程度が大きい割当の科目を変更する科目 R として選ぶ。またはユーザが選択する。

2) 科目 R と関係のある科目を2科目間関係表から全て探す。

3) 制約違反を減少させる変更方法を、次の3通りからユーザが選択する。
・科目 R を別のコマへ移動
・科目 R を解除し、別科目を割り当てる
・科目 R と別の割当の科目と交換
ここで科目は、関連科目全てとする。

4) システムは、それぞれの候補の総合違反点数(変更後)を計算し、色分けして表示する。

5) システムは、最少の総合違反点数(変更後)の候補を選ぶ。またはユーザが選択する。

4 実行例

本システムの実行例を図6,7,8に示す。

図6は、手動編成において科目を選択し、時間帯に割り当てを行っている例である。各時間帯には、その時間帯に割り当てるときの違反点数が表示され、違反の程度(0点が白、10点未満が青、10点以上が赤)によって色分けて表示される。また、その時間帯でボタンをクリックすれば、制約違反の内容を表示する(図6の中央上部のウィンドウ)。

図7は、半自動編成において変更方法が移動のときの例である。この図の各時間帯には、その時間帯に移動したときの総合違反点数が右側に表示され、改善の程度(10点以上改善が白、1点以上10点未満改善が青、改善されないが赤)によって色分け表示される。

図8は、過去の時間割を初期値としてシステムに入力し、本システムで得られた時間割である。入力した過去の時間割は、担当教官の変更や科目の統廃合などにより、所々現在のカリキュラムに違反している。当初、総合違反点数は129点であったが、本システムにより編成後は28点にまで減少した。このように、制約違反は少なくなり、現実に即した時間割が編成された。なお、図8の科目の右側の数値は、その科目の違反点数である。

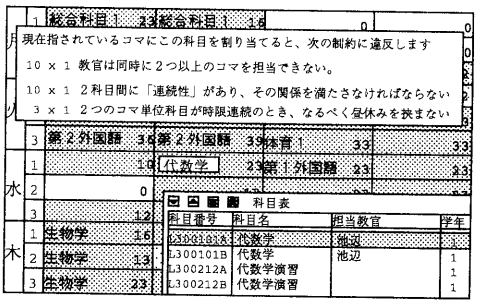

図 6: 手動編成時の表示画面 (部分) の例

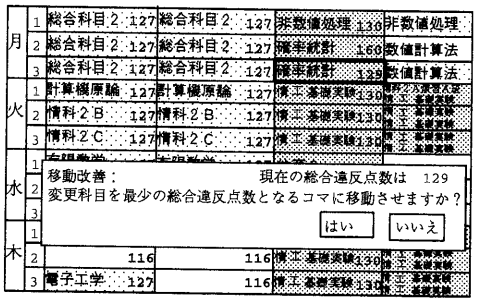

図 7: 半自動編成時の表示画面 (部分) の例

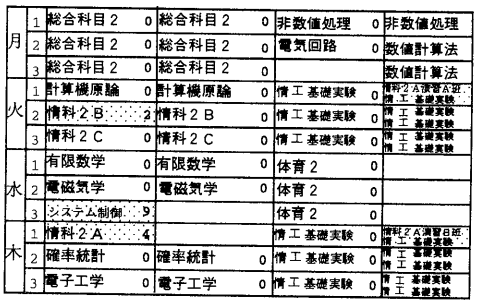

図 8: 本システムで得られた時間割の例 (部分)

5 おわりに

本稿では、従来の自動探索では対処しきれない時間割編成問題を効率よく解くために、対話操作と探索を組み合わせたシステムを開発し、その手法について述べた。

本システムは次の特徴がある。

① 制約の定式化が困難である時間割編成問題において、システムとユーザが協調して探索することによって、システムが扱うことの難しい制約を編成に反映させることができる。

② 制約違反点数を最少化する戦略による山登り法を用いることによって、時間割編成を効率的に行うことができる。

今後は、別の探索手法である Simulated Annealing 法を時間割編成問題に適用することで、半自動編成をより自動化して、さらに実用的なシステムに改良する予定である。

参考文献

[1] 西原 清一: 整合ラベリング問題と応用, 情報処理学会誌, Vol.31, No.4, pp. 500–507 (1990).

[2] T.B.Cooper, J.H.Kingston: The Solution of Real Instances of the Timetabling Problem, *The Computer Journal*, Vol.36, No.7, pp. 645–653 (1993).

[3] 吉田, 窪田, 狩野, 西原: 拡張制約表現による時間割編成システム, 情報処理学会第 48 回全国大会 6N-5 (1994).

[4] S.Minton, M.D.Johnston, A.B.Philips, and P.Laird: Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. , *Artificial Intelligence*, Vol. 58, pp. 161–205 (1992).

[5] 西森, 内野, 狩野, 西原: 制約違反最少化戦略による対話型時間割編成システム, 情報処理学会第 50 回全国大会 6P-6 (1995).

[6] B.Selman, H.Levesque, D.Mitchell: A New Method for Solving Hard Satisfiability Problems, *AAAI-92*, pp.440–446(1992).

[7] P.Morris: The Breakout Method For Escaping From Local Minima, *AAAI-93*, pp. 40–45(1993).

付録

3.1 で記載しきれなかった制約の制約充足判定の内容を以下に収録した。

(1) 科目単位制約 C_U

表 3-b: 教官の担当不可能時間帯表

教官名	担当不可能時間帯
井田	(1, *, *)
大田	(*, [月, 水], *)
斉藤	(*, *, [3,4,5])
⋮	⋮

表 3-c: 教官の優先担当時間帯表

教官名	優先担当時間帯
池辺	(*, *, [3,4,5])
斉藤	(*, [月, 火, 水], *)
田中	([1,2], *, *)
⋮	⋮

c_2 : 科目 i の担当教官 $teacher(i)$ が表 3-b に入っているとき、その担当不可能時間帯に科目 i の割当時間帯が入っていれば、制約違反。

c_3 : 科目 i の担当教官 $teacher(i)$ が表 3-c に入っているとき、その優先担当時間帯に科目 i の割当時間帯が入っていなければ、制約違反。

(2) 2 科目間制約 C_R

c_6 : 科目 k_1, k_2 の関係が学期集中であるとき、かつこれらの関係が時限連続でないとき、割当時間帯が同じ曜日ならば、制約違反。

c_7 : 科目 k_1, k_2 の関係が学期連続であるとき、これらの割当時間帯が同曜日同時限でないならば、制約違反。

c_8 : 科目 k_1, k_2 の関係が時限連続であるとき、これらの割当時間帯が昼休みを挟むならば、制約違反。

(3) 全体制約 C_E

c_{10} : 任意の科目 $i_1, i_2 (i_1 \neq i_2)$ に対して、$course(i_1) = course(i_2)$、かつ $slot(i_1) = slot(i_2)$、かつ $grade(i_1) = grade(i_2)$ のとき制約違反。

c_{12} : 全ての科目 i $(i = 1, \ldots, M)$ に対して、$slot(i)$ が等しいものの個数が 2 個ならば制約違反。

c_{13} : 任意の科目 $i_1, i_2 (i_1 \neq i_2)$ に対して、$course(i_1) = course(i_2)$ = 必修、かつ、$grade(i_1) \neq grade(i_2)$、かつ、$slot(i_1) = slot(i_2)$ のとき制約違反。

c_{14} : 全ての科目 i $(i = 1, \ldots, M)$ に対して、$grop(i)$ が等しく、$slot(i)$ の学期・曜日が等しいものの個数が 4 個以上ならば制約違反。