# 学校時間割り自動編成システム

NEC C&C メディア研究所
吉川 昌澄 (yosikawa@ccm.cl.nec.co.jp)

## 1 はじめに

各高校では毎年春休みに次年度の毎週の時間割りを編成している．大規模な学校では，この作業は極めて困難であり，150 人日もの労力を掛けているのが実情である．また，大学や予備校など，各種の学校では時間割り編成のために 2 ～ 3 人程度の要員を配置している例もある．このように，時間割りの編成は極めて困難な作業であり，その自動化が望まれている [PATAT 96, PATAT 98].

筆者らは，これまでに，学校時間割り編成問題を制約充足問題 (CSP: Constraint Satisfaction Problems) として定式化し，汎用的なアプローチで自動解決技術を開発してきた．現在では，中学校，高等学校，大学向けのソフトウェアパッケージとして広く販売され利用されている．

一般に制約充足問題は NP 完全であり万能の解法は存在しない．そこで高速の近似解法やユーザ編集機能が重要となる．また，現実世界の問題を如何に定式化し，不完全な解法を用いてシステム化するかというモデリングの問題が極めて重要となる．

本稿は，筆者らが開発した学校時間割り自動編成技術を概説すると共に，汎用的制約問題解決技術の重要性について述べる．以下では，まず，時間割り編成問題の定式化，続いて，問題のモデリング，問題解決手法について概観し，学校時間割り編成システムと大学時間割り自動編成について述べる．最後に，今後の課題と展望について考察する．

## 2 学校時間割り編成問題の定式化

ここでは，まずモデルとなる制約充足問題について，続いて学校時間割り編成問題について述べる．

### 2.1 制約充足問題 (CSP)

CSP は，有限個の変数 $v_1,...,v_N$ と，各変数の取り得る値の有限集合である候補集合 $D_i = \{x_1^i,...,x_{d_i}^i\}$ と，有限個の制約 $C_1,...,C_M$ とからなる．制約 $C_j$ は，その制約に関与する変数の組 $\{v_{k_1},...,v_{k_{M_j}}\}$ と，関与変数の値が満足すべき論理式 $C_j^{k_1,...,k_{M_j}}(v_{k_1},...,v_{k_{M_j}})$ を持つ．ここで，1 変数に関与する制約をユナリ (**unary**) 制約，2 変数の場合バイナリ (**binary**) 制約，一般に **N** 変数間 (*N*-**ary**) 制約と呼ぶ.). 問題は，すべての制約 $C_j$ を同時に充足するような各変数 $v_i$ の値 $x_l^i$ を対応する候補集合 $D_i$ 中から選択する組み合せ探索問題である [吉川 98].

CSP の例として三色問題がある．これは，日本の都道府県区分図のように，平面上の線により区切られた各領域を，隣接する領域が同色にならないように三色で塗り分ける問題である．各領域の色を変数 ($v_1$ : 東京, $v_2$ : 神奈川,...), 三色の集合をすべての変数の候補集合 ($D_i = \{$赤,緑,青$\}$), 各隣接領域の組の間で同色とならないという条件を制約 ($C_1^{東京,神奈川}(x,y) = x \neq y$; $C_2^{東京,千葉}(x,y) = x \neq y$; ...) とすれば，制約充足問題として定式化可能である．その他のトイプロブレムとして，N-Queen 問題 [Minton 92], SAT 問題，各種グラフ問題，パズルなどがある．

### 2.2 学校時間割り編成問題

| | 1-1 | ⋯ | 3-10 |
|---|---|---|---|
| 月 1 | 数 I | ⋯ | 古文 |
| ⋯ | ⋯ | ⋯ | ⋯ |
| 土 4 | 現文 | ⋯ | 幾何 |

(a) クラス時間割り表

| | 田中 | ⋯ | 佐藤 |
|---|---|---|---|
| 月 1 | 数 I | ⋯ | 古文 |
| ⋯ | ⋯ | ⋯ | ⋯ |
| 土 4 | 幾何 | ⋯ | 現文 |

(b) 先生時間割り表

図 1: 高校時間割り編成問題のイメージ

高校の時間割り編成問題 (図 1) は，1 週間の授業や会議の予定を決定する問題である．クラス数 30・先生 70 人程度の大きな学校では，延べ 100 ～ 150 人日という工数を掛けて時間割りを編成している．各授業に出席する先生や対象クラス，利用する特殊教室などの属性はあらかじめ与えられる．ここでは，各授業を開催する曜日時限が CSP の変数となり，1 週間の時限 $\{$月1,...,土4$\}$ が候補集合となる．主な制約は次の通り．(1) 同じクラス / 先生の授業が同じ時



© 日本オペレーションズ・リサーチ学会．無断複写・複製・転載を禁ず．

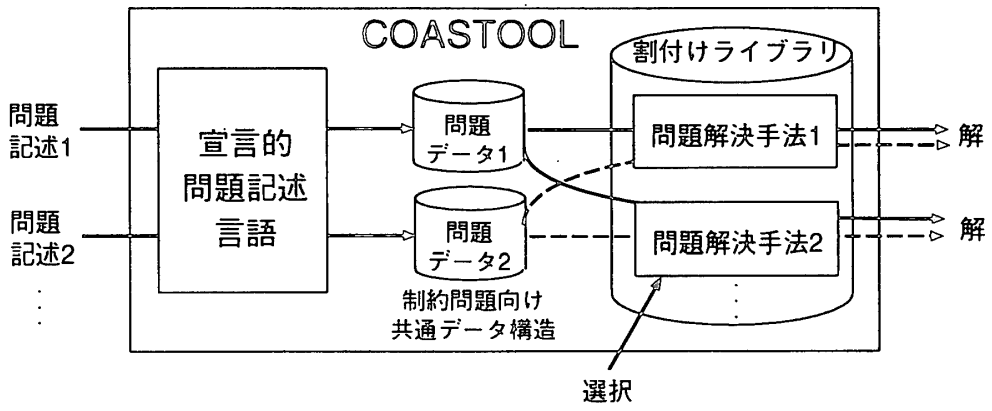


図 2: COASTOOL アーキテクチャ

```
(def-set  授業   数 I1-1a  数 I1-1b …)    ;; 変数の集合
(def-set  時限   月 1  月 2 … 土 4)        ;; 候補集合
(def-constraint  非常勤田中先生の都合          ;; 制約
  :variables    ((v  田中授業))            ;; 関与変数
  :condition    (not (is-a  v  田中不在時間))  ;; 条件
  :penalty      10)                       ;; 違反点数
(def-problem  時間割り編成問題                 ;; CRP
  :variables    ((授業  時限))          ;; 変数と候補集合
  :constraints  (非常勤田中先生の都合 …)   ;; 制約を列挙
  :minimize     制約違反点数の総和)         ;; 目的関数
```

(a)COASTOOL/Lisp

```
class TeacherAbsence :public Constraint {
   Teacher *_tea;
   TimeSlotVariable *_var;
 public:
   TeacherAbsence(Teacher *t, Lesson *l)
      :_tea(t), _var(l->timeSlot()) {} ;
   virtual int doCheck() const {
      return(isTeacherAbsentAt(*_tea, _var->value())) } ;
   virtual float penalty() const { return 10.0 ;};
   ... }
```

(b)COASTOOL/C++

図 3: COASTOOL による問題記述例

限に重ならない. (2) 選択授業など同時開催の授業や, 芸術など連続開催の授業がある. (3) 特殊教室での同時開催授業数の制限. (4) 先生などの予定により不都合な時限がある. (5) 各クラスの同じ教科 / 科目の授業は異なる曜日にする. (6) 先生の 1 日 / 連続の授業数の制限. (7) 週 2 駒の授業はなるべく中 1 日以上空ける.

ここで問題となるのは, 例えば, 制約 7 に「なるべく」とあるように, 絶対とは言わないが充足が望ましいという制約があることである. 現に実際の高校では一部の制約を違反した時間割りが運用されている.

このように, 制約を違反不可能なものと可能なものとに分類した場合, 前者を絶対制約 (または強制約), 後者を考慮制約 (または弱制約) と呼ぶ. 高校によっても異なるが, 制約 5 ～ 7 が考慮制約であろうか.

このように絶対制約と考慮制約を持つ問題を定式化する枠組みとして, CSP を若干修正した制約緩和問題 (CRP) を定義する. CRP は, CSP のモデルに加えて各制約が違反時の罰則となる違反点数を持つ. 問題は, 違反している制約の違反点数の合計が最小となるような割り付け (各変数とその値の組) を探索する組合せ最適化問題である. 絶対制約と考慮制約がある場合, 絶対制約には十分大きい違反点数を, 考慮制約にはその重要度に応じた違反点数を与える事により, 現実問題の要求に則した定式化が可能となる.

## 3 学校時間割り編成問題のモデリング

筆者等は, 制約ベース計画シェル COASTOOL (COnstraint-based Assignment & SchedulingTOOL)[吉川 93, Yoshikawa 94] を開発し, 各種時間割り編成問題や生産計画問題に適用して来た. ここでは, COASTOOL を中心に, 現実問題のモデリングについて述べる.

### 3.1 COASTOOL/Lisp によるモデリング

COASTOOL は汎用的なスケジューリング問題解決を目指すシェルである. 前章で述べたように, 制約問題は現実の時間割り問題のモデルとして有効である. しかし, その一方で, 制約問題は NP 完全であり, 無二万能の解法は存在



© 日本オペレーションズ・リサーチ学会. 無断複写・複製・転載を禁ず.

しないと言う課題がある．そこで，COASTOOL は制約問題モデルに基づく宣言的問題記述言語と，ある範囲の問題に有効な汎用的解法を集めた割り付けライブラリとから構成される (図 2)．ユーザは現実問題を制約問題に定式化し，宣言的問題記述言語を用いてこれを記述し (図 3(a))，その問題に相応しい解法を選択することにより問題を解決できる．ここで重要な点は，問題自身とその解決方法が独立であることである．したがって，1 つの問題に異なる解法を適用すること，また逆に，異なる問題に 1 つの解法を再利用することが可能である．また，適当な解法がない場合には，割り付けライブラリの制約伝播等の機能を用いて，個別の解法を開発する．

筆者等は，COASTOOL/LISP を用いて久喜北陽高校の時間割り問題をモデル化した [Yoshikawa 94]．まず高校を訪問し 3 時間程度のインタビューでデータを入手，1.5 人日の工数で問題 (制約 500 行，データ 1,000 行) を記述した．COASTOOL が編成した時間割りを先生にファックスしてコメントを頂き，データの修正，制約の追加・削除・違反点数の調整などのフィードバックを 10 回程度繰り返した．これにより，わずか 3 人日程度で人手並みに高品質の時間割りを自動編成することに成功，エキスパートシステムのエンジン部分の開発を完了した．この成功の要因としては，問題と解法が独立であり解法のヒアリングや実装が不要なこと，宣言的な制約問題のモデルによりモデル化が容易なこと，フィードバックが容易で緻密なプロトタイピングが可能なことなどが考えられる．

## 3.2 COASTOOL/C++ による制約の実装

スケジューリングシステムはデータベース (DB) 上のデータをメモリ上の制約問題データ構造に展開し，解法アルゴリズムにより自動立案を行い，画面インタフェース (GUI) を通してユーザ編集を行う．一般に制約問題は NP 完全のため，メモリ上での効率的なデータの実装方法，高速かつ高品質な解法アルゴリズムが重要となる．COASTOOL/LISP には，問題の記述能力，メモリ容量，速度性能，DB や GUI との連携などの課題があり，COASTOOL/C++ を開発した [Yoshikawa 96]．COASTOOL/C++ は一種のオブジェクト指向フレームワークで，宣言的問題記述言語の代わりに，変数や制約など制約問題解決に必要なクラスライブラリを提供する．ユーザはクラスライブラリを用いて，メモリ上に制約問題データを構築し (図 3(b))，割り付けライブラリの解法を適用する．(または，ライブラリを用いて個別解法を開発する．)

COASTOOL/C++ の最大の特長は，仮想制約と呼ぶ制約管理方法である．例えば，制約 1「同じ先生の授業が同じ時限に重ならない」(2.2節) を考える．今，ある先生が 3 つの授業 x, y, z を持つとしよう．この制約には，2 通りの実装方法が考えられる．一つは，この先生が受け持つ相異なる 2 つの授業の組み合わせすべてに対して，同じ時間にならないというバイナリ制約 ($C_1^{x,y}(x,y) = x \neq y$; $C_2^{y,z}(y,z) = y \neq z$; $C_3^{z,x}(z,x) = z \neq x$) を定義する方法である．もう一つは，この先生が受け持つすべての授業の組に対して，重なりがないという一つの制約 ($C_0^{x,y,z}(x,y,z) = (x \neq y) \wedge (y \neq z) \wedge (z \neq x)$) を定義する方法である．前者は制約の個数が組み合わせ的に爆発するという課題がある (一般に先生が $n$ 個の授業を持つ場合は ${}_nC_2$ 個)．後者は制約違反時にどの変数を直せばよいか不明瞭であるという課題がある．そこで，両者を組み合わせて，それぞれの利点を利用可能にしたのが仮想制約である．すなわち，常に後者の形の制約 1 個 ($C_0$) を持ち，制約違反が発生した場合にはその部分に関して動的に前者の形の制約 ($C_1, C_2, C_3$) を生成する．仮想制約により，制約の個数を制限し，かつ，制約違反時に変更すべき変数が明らかになる．

例として，制約 6「先生の 1 日授業数の制限」を考える (COASTOOL/LISP では実装されていない)．ある先生の授業が 15 個あり 1 日授業数の制限が 4 だとすると，違反する変数の組み合わせは ${}_{15}C_5 = 273$ 通りある．この大量の制約も，1 つの仮想制約と違反している制約の実態により置き換えることができる．COASTOOL/C++ では，仮想制約を用いることにより，メモリ容量を 1/10 に削減，立案時間を 1/20 に短縮することに成功した [Yoshikawa 96]．仮想制約のメカニズムは複雑であり，その実現は決して簡易ではない．しかし，さまざまな制約をシステム上に実装し効率的に処理するためには，このような実現方法の工夫が必要である．

## 4 学校時間割り編成問題の解法

主な CSP の解法には，無矛盾性 (consistency) 手法，バックトラック法，局所探索手法がある．

学校時間割り編成問題において，1 週間の時限数 34，クラス数 30 の高校では，授業の全駒数は約 1,000 弱となる (選択授業など複数クラスにまたがる授業がある)．したがって変数は約 1,000 個の授業，候補集合は 34 の時限となり，その組合せの数は $34^{1000} \simeq 10^{1531}$ となる．このような膨大な探索空間において，無矛盾性手法，バックトラック法 (あるいは分枝限定法) は計算量爆発の問題がある．また，制約緩和の可能性があるため，枝刈りによる探索空間の限定は困難である．

そこで近年注目されているのが局所探索法である．局所探索法は，まず，乱数などを用いて初期割付を作成する．通常，初期割付は何らかの制約を違反している．次に，制約違反が小さくなるように (または目的関数を改良するように) 繰り返し改良操作を施し，解または近似解を求める．解発見の完備性がない反面，制限時間内に何らかの近似解を得ら



© 日本オペレーションズ・リサーチ学会．無断複写・複製・転載を禁ず．

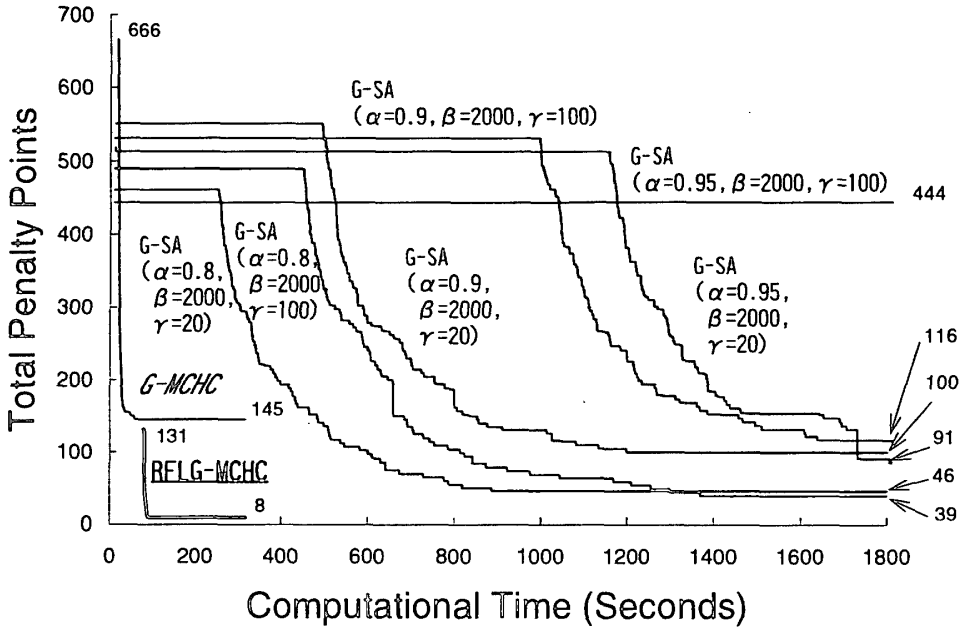


図 4: 高校時間割り編成の実験結果: G は Greedy 初期割付. SA で $\alpha$ は冷却比率, $\beta$ は同温での繰り返し回数, $\gamma$ は初期温度. プロットは初期割付終了以降のみ.

れる点, 制約緩和が可能な点などが時間割りへの応用に適している.

改良操作の内容や繰り返しの制御方法により様々な方法がある. 山登り法は, 改悪となる改良操作を行わない手法である. **MCHC (min-conflicts hill-climbing)** 法 [Minton 92] は, 制約を違反する変数をランダムに選び, その変数の値を違反制約の個数 (または違反点数) が最小になる値 (同点時はランダム) に変更する. MCHC 法は, 最適性は低いが大規模問題の高速解決に適している.

**SA (simulated annealing)** 法 [Nakakuki 94] は, 金属の焼なましを模倣したランダムに改悪する局所探索法である. 制御変数に温度を設け, 高温から徐々に冷やしていく. 高温時ほど高い確率で改悪を受け入れる. ゆっくり冷やせば初期割付によらず最適解に到達するが, かなりの計算時間を要する. 速度性能よりも最適性を追求する問題に適している.

筆者等は高校時間割り編成にバックトラック法を適用したところ, 数日計算機を走らせ続けても最初の解には到達できなかった. MCHC 法は高速だったが先生の手作業の 2/3 程度の品質であった. SA 法は品質は少し良いが計算時間が長かった.

そこで, 高品質の初期割付法である **RFLG (Really Full Lookahead Greedy)** 法 [Yoshikawa 94] を開発し MCHC 法と組み合わせたところ, 1 ~ 2 分で先生の手作業の 90% 以上の品質の時間割りを自動編成することに成功した (図 4).

**RFLG** 法は, 制約伝播により候補集合の絞り込みを行い, 残った候補を順次変数に割り付ける. 割り付けの結果は, 再度, 制約伝播で残りの変数へ伝播される. 候補数の少ない変数と他の変数への影響の小さい値を優先している. 途中, 候補が無くなった変数は後回しにして, 違反点数が最小となる値を順次割り付ける **Greedy** 初期割付法 により割り付けを行う. 制約伝播を用いることにより, MCHC 法の結果よりも優れた初期割付を作成できた (図 4).

ここで用いている制約伝播はアーク無矛盾性 **(AC: arc consistency)** 手法 [Mackworth 92] である. これは, 個々の制約を考えて, 充足の可能性のない値を候補集合から取り除く処理である. 例えば, 制約 $x < y$ について, 候補集合 $D_x = D_y = \{1,2,3\}$ を絞り込めば $D_x = \{1,2\}$. $D_y = \{2.3\}$ となる. いずれの制約伝播も, 何らかの変化が生じた場合には, 最終的に変化が無くなるまで伝播を繰り返す.

また, その後, RFLG を改良し, 局所最適解脱出手法を開発した. RFLG の改良は候補が無くなった変数を後回しにせず, その時点で繰り返し改良を適用する段階的立案手法である [金子 97a]. 局所最適解脱出手法は, 同じクラスの授業が同じ時限に重ならない (制約 1) を利用して. 同じクラスの 2 つの授業を玉突きのように動かすことにより, 局所



© 日本オペレーションズ・リサーチ学会. 無断複写・複製・転載を禁ず.

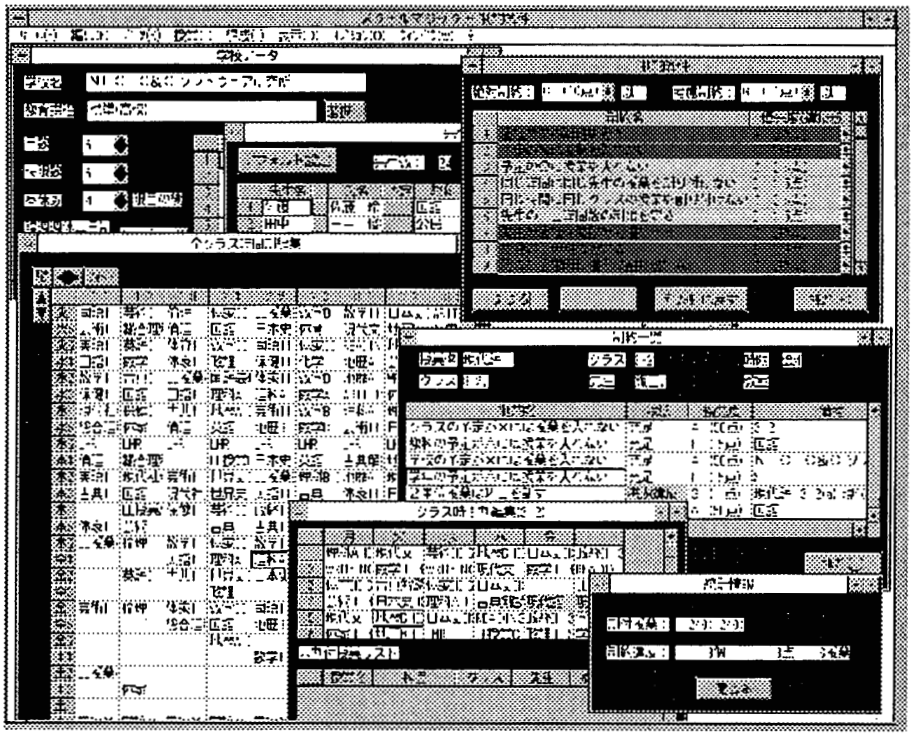

図 5: スクールマジック画面イメージ

最適解から脱出する時間割り専用の方法である [金子 95]. これらの改良により, 違反点数をさらに 30 ~ 97% 削減することに成功し, より制約の厳しい高校に対しても, 先生の人手の時間割りに遜色の無い品質を得ることが可能となった [Kaneko 99].

## 5 AI 学校時間割り自動編成システム「スクールマジック」

図 5に, AI 学校時間割り自動編成システムスクールマジックの画面イメージを示す. スクールマジックは, 各種データ編集, 制約違反点数の設定, 各種時間割り表示, 制約違反状況や推薦移動先の提示など, 約 40 種類の画面を備えるパッケージソフトウェアである.

スケジューリング問題における違反点数や目的関数を正確に定めることは難しい. また, 厳密な定式化ができたとしても, 解法の能力が不十分な場合も多い. そこで, GUI(図 5) を通したユーザ編集機能が重要となる. ここでは**主導権混在型 (mixed-initiative) 問題解決法** [Yoshikawa 96](図 6) が有効である. 例えば, 自動割付の結果をユーザが確認し一部をフィックス, 一部を修正し, その結果を初期割付とみなして繰り返し改良を適用する. このように, システムの自動機能とユーザの手動機能を自由に行き来する協調的な問題解決が可能である. 局所探索法は, 再計画同様, 自動修正にも有効である.

情報提示機能としては, 制約の違反状況の表示, 修正を施した場合の影響を示す what-if 解析, 最適な修正方法を示す推薦機能, 割付不可能な理由を解析する説明機能などがある. これら機能を実装する上で, システムが制約問題を宣言的なデータとして保持することが有効である. また, システムとユーザが同一の問題データを対象とすることが重要となる. 例えば, ユーザが制約や目的関数を修正した場合, システムの自動機能が機敏に対応可能なことが望ましい.

結局, スクールマジックにより, 従来 150 人日を要して時間割りを編成していた高校でも, データ入力から開始して 2 人日程度で時間割りを完成することが可能となった. スクールマジックは; 現在, 100 を越える中学, 高校で利用されている [スクールマジック].

## 6 大学時間割り自動編成システム

筆者等は, 大学版スクールマジックを開発, 早稲田大学の時間割り編成に適用し, 実用化に成功した [金子 97b]. 大学時間割り編成問題は, 教員があらかじめ決められた各講義や会議を, さまざまな制約を考慮しながら, より多くの制約を満足するように 1 週間のうちの一つの時限に割り付けると同時に, 講義を行なう教室に割り付ける問題である. 大



© 日本オペレーションズ・リサーチ学会. 無断複写・複製・転載を禁ず.

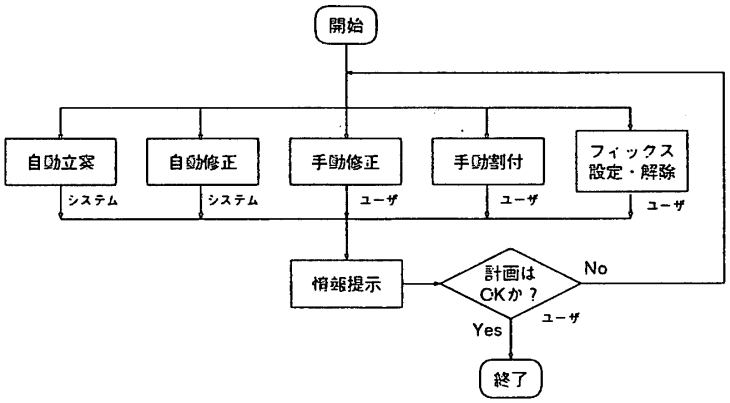


図 6: 主導権混在型問題解決のフロー

学時間割り編成問題は高校時間割り編成問題と類似であるが, 下記の点で異なっている. (1) 高校時間割りに比べて問題規模が大きい. (2) 時限だけでなく教室の割り付けが必要. (3) 通年と前期のみと後期のみの講義が混在し前期と後期の時間割りを同時に編成する. (4) 講義は選択性のため基本的にクラスはない. しかし, 同一専修コースの必須科目など, 重ねてはいけない講義がある. (5) 先生の希望教室・時限や昨年度の時間割りの割り付け情報を元に類似の割り付けを優先する.

| 初期割り付け手法 | 一部割り付け情報あり | | | 割り付け情報なし | | |
|---|---|---|---|---|---|---|
| | 違反点 | 時間 (秒) | 最良結果の回数 | 違反点 | 時間 (秒) | 最良結果の回数 |
| Greedy | 399.2 | 193.7 | 0 | 96.9 | 178.1 | 0 |
| RFLG | 395.1 | 3349.2 | 0 | 8.5 | 678.4 | 6 |
| 段階 RFLG | 0.9 | 704.0 | 7 | 26.5 | 837.6 | 6 |

図 7: 大学時間割り編成での実験結果

筆者等は, COASTOOL/C++ 上に大学時間割り編成問題を定義し, 高校時間割りとまったく同一の割り付けライブラリをそのまま適用し, 実用化に成功した. すなわち, 10 分程度の計算時間で, 人手並みの高品質な時間割りの自動立案に成功した. これは, COASTOOL/C++ アーキテクチャの汎用性が有効であることを示している. 解法の点では類似の特性を示したが, 昨年度割り付け情報利用時は, 特に, 段階的立案手法が有効であった (図 7). これは, 段階的立案手法が昨年度割り付けの問題点を早期に解消してから初期割り付けを進めるためである.

## 7 今後の課題と展望

上記のように, 筆者等は COASTOOL による汎用的制約問題解決により, 逐次改良手法を適用することにより, 中学・高校・大学向け時間割り自動編成システムスクールマジックを開発し, 100 以上の学校で利用されるに至っている.

まず重要な点は, 大規模複雑な現実の時間割り問題に対し, 制約伝播を用いた高品質初期割り付けと高速の繰り返し改良手法の組合せによる解法が, 極めて高い有効性を示したことである. 従来の汎用的解法技術に比べて, 飛躍的な高速性と高品質性を同時に満足し, 実用的にも人手とほぼ同等の時間割りの自動編成を可能とした.

次に重要な点は, COASTOOL において問題のモデリングと解法とを分離したことである. これにより, 微妙で複雑な現実問題の精緻な定式化が可能となった. また, 同一の割り付けプログラムが中学・高校・大学向けそれぞれのスクールマジック製品において利用されるに至った.

しかし, 特に実用に耐える COASTOOL/C++ での制約や制約問題の記述には, 研究者レベルの専門的能力が要求される. このため, 個々の問題に対して, 膨大な開発努力が必要となるのが現状である.

そこで, 筆者等はエンドユーザレベルでの制約問題記述を可能とする GUI ツールとして制約データシートを試作し, 時間割り問題の簡易記述に成功した [吉川 99a, 安藤 99, 吉川 99b]. 制約データシート (図 8) は, リスト・二次元表など構造を持つデータシート間の演算と制約を基本とする新しい表計算モデルを持ち, スプレッドシートの簡易性



© 日本オペレーションズ・リサーチ学会. 無断複写・複製・転載を禁ず.

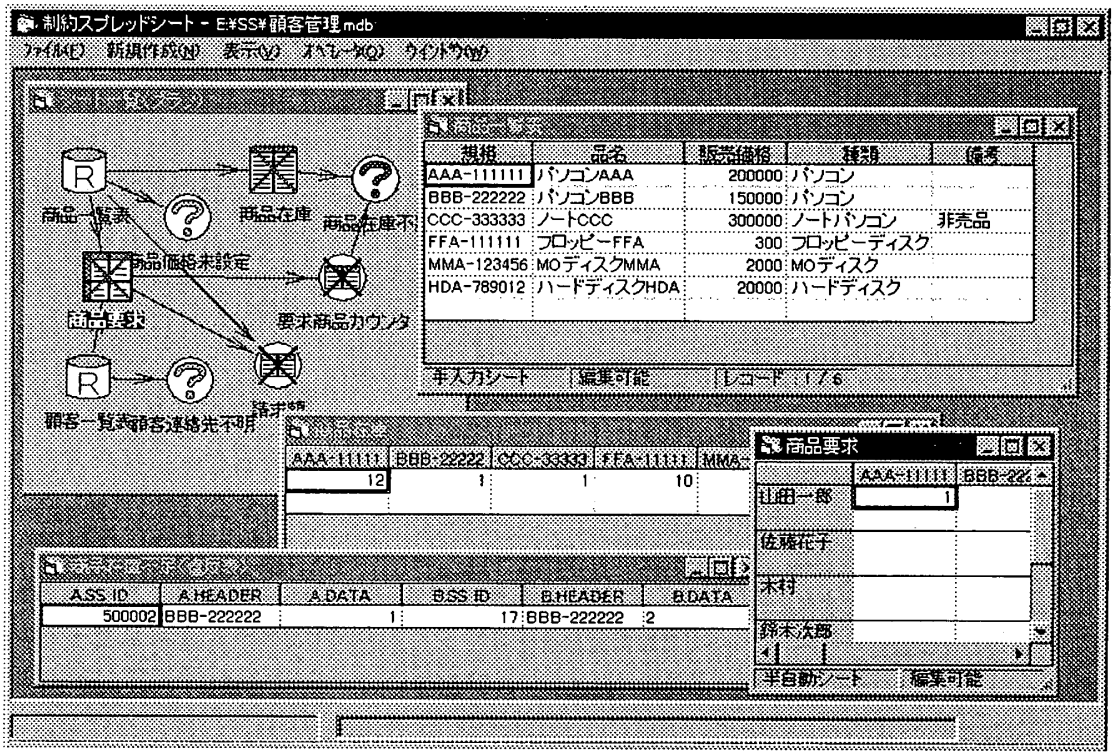


図 8: 制約データシート画面イメージ

とデータベースの頑強性や再利用性を合わせもつ表計算ツールである．構造変換・フィルタ・セル値演算などデータシートの構造に基づく新しい表演算系と，表間演算処理の自動再計算機構および一時的な違反を許す柔軟かつ強力な制約管理を特徴とする．特に制約違反表と呼ぶ RDB 表形式のシートを定義することにより，制約条件を定義可能である．

制約データシートによりスクールマジックの制約問題を定義したところ，すべての制約条件の簡易記述に成功した． 図 9に「先生の 1 日授業数の制限」(制約 6) の記述例を示す． 全制約条件の記述に要した記述量は，COASTOOL/C++ で約 1,000 行であるのに対して，制約データシートでは約 50 のシート演算オペレータで記述可能である (図 10)．また，WYSWYG インタフェースにより簡易に記述可能であり，SE レベルの能力を持たない一般エンドユーザでも制約問題の記述が可能となることが期待される．

今後，制約データシートで記述された制約問題から COASTOOL/C++ レベルの効率的実装方法にコンパイルするコンパイラの開発を進める予定である． これが完成すれば，制約データシートのバックエンドに COASTOOL/C++ の割り付けライブラリを与えることにより，汎用制約問題解決パッケージツールが実現可能となろう．これは，現在の数理計画パッケージと類似だが，記号処理による制約が可能であり，学校時間割り自動編成システムや要員業務割り当てシステム等を簡易に構築可能なツールである．

## 8 おわりに

本論文では，筆者等の時間割り自動編成システム開発を概観し，汎用的なアプローチによる制約問題解決への試みについて述べた．Galil は，“アルゴリズム設計にレシピはない．これは応用面では不運であり毎回いちから設計しなければならない．しかし，研究者は幸運であり研究テーマが無くなることはない．” と述べている．NP 完全性を考慮すれば，これは自然な結論であると考えられる．これをビジネスの観点から見れば，市場規模の大きい個別問題に対する個別問題向け解法の研究は価値があるが，小さいものはペイしないという結論になる．今後，汎用制約問題解決パッケージツールの実現により，時間割り問題や要員業務割り当て問題など，多種多様の問題が簡易に解決可能となることを期待する．

## 参考文献

[安藤 99] 安藤友人，吉川昌澄: “制約データシート (2): システムの試作と評価”，第 *13* 回 *AI* 全大 (1999).



© 日本オペレーションズ・リサーチ学会．無断複写・複製・転載を禁ず．

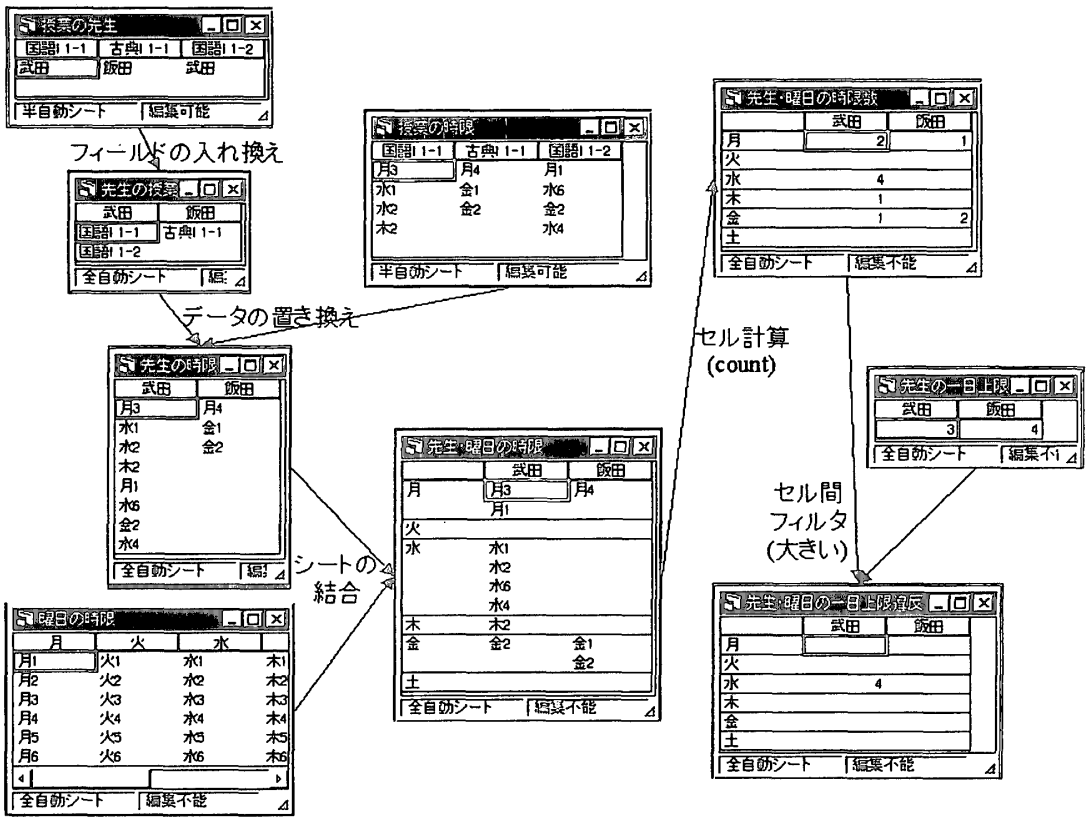


図 9: 制約データシートによる制約定義例


[PATAT 96] E. K. Burke and P. Ross (eds.), *The Practice and Theory of Automated Timetabling*, Springer-Verlag Lec. Notes in CS., 1153, 1996.

[PATAT 98] E. K. Burke and M. W. Carter (eds.), *The Practice and Theory of Automated Timetabling Volume 2*, Springer-Verlag Lec. Notes in CS., 1998 (To appear).

[金子 95] 金子, 吉川, 山之内, 渡辺, "学校時間割自動編成システムにおける制約緩和手法の提案と評価," 第 9 回 AI 全大, 1994, pp. 483–486.

[金子 97a] 金子, 吉川, 山之内, 渡辺, "学校時間割編成問題における段階的編成手法の提案と評価," 第 54 回情処全大, 2:321–322, 1997.

[金子 97b] 金子, 吉川, 山之内, "大学時間割編成問題への制約緩和手法の適用と評価," 第 55 回情処全大, 2:578–579, 1997.

[Kaneko 99] K. Kaneko, M. Yoshikawa, and Y. Nakakuki, "Improving a Heuristic Repair Method for Large School Timetabling Problems," *Proc. 5th Int'l. Conf. on Principles and Practice of Constraint Programming (To appear)*, October 1999.

[Mackworth 92] A. K. Mackworth, "The Logic of Constraint Satisfaction," *Artif. Intell.* 58:3–20, 1992.

[Minton 92] S. Minton, M. D. Johnston, A. B. Philips, and P. Laird, "Minimizing Conflicts: A Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems," *Artif. Intell.* 58:160–205, 1992.

[Minton 96a] S. Minton, "Specification-By-Demonstration: The ViCCS Interface," *AAAI Spring Symposium Seriese*, 1996.

[Minton 96b] S. Minton, "Automatically Configuring Constraint Satisfaction Programs: A Case Study," *Constraints* 1(1), 1996.

[Nakakuki 94] Y. Nakakuki and N. Sadeh. "Increasing The Efficiency of Simulated Annealing Search by Learning to Recognize (Un)Promising Runs," *Proc. 12th Nat. Conf. on AI*, 2:1,316–1,322, 1994.

[スクールマジック] NEC ソフトウェア, スクールマジックホームページ, *http://www.necsoft.co.jp/soft/smagic.html.*

[吉川 93] 吉川, 潮田, 渡辺, "制約ベース計画シェル COAST の開発と評価," 第 7 回 AI 全大, No. 14-7, 1993.




© 日本オペレーションズ・リサーチ学会. 無断複写・複製・転載を禁ず.

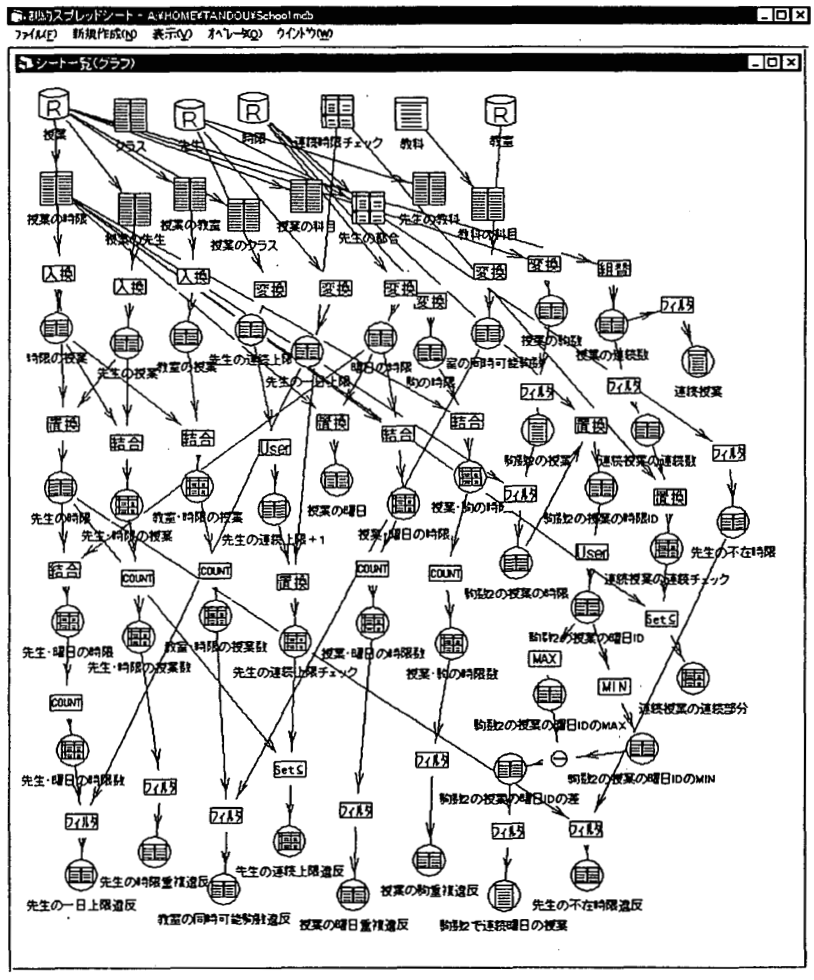

図 10: 学校時間割り編成問題の全制約の記述


[Yoshikawa 94] M. Yoshikawa, K. Kaneko, Y. Nomura, and M. Watanabe, "A Constraint-Based Approach to High-School Timetabling Problems: A Case Study," *Proc. 12th Nat. Conf. on AI*, 2:1,111–1,116, 1994.

[Yoshikawa 96] M. Yoshikawa, K. Kaneko, T. Yamanouchi, and M. Watanabe, "A Constraint-Based High-School Scheduling System," *IEEE Expert*, 11(1):63–72, 1996.

[吉川 98] 吉川昌澄, "制約最適化技術のスケジューリング問題への応用", 人工知能学会誌 *Vol. 13, No. 3,* pp.43–50 (1998).

[吉川 99a] 吉川昌澄, 安藤友人: "制約データシート (1): シート間制約に基づく新しい表計算モデルの提案", 第 *13* 回 *AI* 全大 (1999).

[吉川 99b] 吉川昌澄, 安藤友人: "制約データシートによる制約問題記述", 第 *44* 回 *AI* 学会知識ベースシステム研究会 (1999).




© 日本オペレーションズ・リサーチ学会. 無断複写・複製・転載を禁ず.