解　説

制約最適化技術のスケジューリング問題への応用

Application of Constrained Optimization for Practical Scheduling Problems

吉川　昌澄*
Masazumi Yoshikawa

* NEC C&C メディア研究所
C&C Media Research Laboratories, NEC.

1998 年 2 月 19 日　受理

Keywords: constraint satisfaction/relaxation/optimization, timetabling, production scheduling.

1. ま　え　が　き

定理証明・計画問題・設計問題など, AI 分野の多くの問題は制約充足問題 (CSP: Constraint Satisfaction Problems) として定式化可能である. その研究内容は多岐にわたり, AAAI-96 においては, 探索制御・学習, 相変遷 (phase transition), 統計的探索など, 9 つの CSP セッションが開設された.

CSP を中心とする制約問題の応用分野も幅広い. スケジューリング分野においては, 実用化システムが多数開発され事業化が進んでいる. 特に生産計画では, 制約最適化に基づく先端生産計画スケジューリングシステムがサプライチェーン管理と呼ばれる急成長市場を開拓している [PeopleSoft, i2]. NASA を中心とした宇宙関係の応用研究も盛んである [JPL]. また, 時間割り編成・要員業務割り当てといった比較的新しい事業分野においても, 基礎・応用の両面から研究が進められている [PATAT 96, PATAT 98].

一般にスケジューリング問題は **NP 完全**[*1]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, 現実世界の問題をいかに定式化し, 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, 筆者の時間割り編成や生産計画の経験を中心に, 応用研究の観点から制約問題解決技術の現状と課題を解説する. 以下では, まず, スケジューリング問題の定式化, 続いて, 問題のモデリング, 問題解決手法について解説し, 今後の課題と展望について考察する.

*1　問題規模の多項式時間の計算量では解決不可能.

2. スケジューリング問題の定式化

制約充足問題 (CSP) はスケジューリングの基礎となるが, 現実問題はその範囲に収まらない. ここでは, CSP を定義し, **制約緩和問題** (CRP: Constraint Relaxation Problem)・**制約最適化問題** (COP: Constrained Optimization Problem) を導入する.

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[*2], 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[*3]. 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**組合せ探索問題**である[*4].

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 (図 1). 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 ($C_1^{\text{東京, 神奈川}}(x, y) =$

*2　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.

*3　論理式の代わりに, M_j 次元配列でも表現可能だが, 規模が膨大になり非実用的である.

*4　派生的に, 解の存在の判定や全解の探索問題もある.

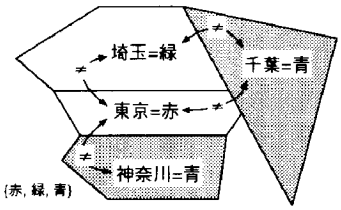

図1　三色問題

表1　制約問題のモデル

		CSP	CRP	COP
変数 v_i		○	○	○
候補集合 D_i		○	○	○
制約 C_j	充足条件	○	○	○
	違反点数	−	○	△
目的関数		−	−	○
応用領域		−	時間割り	生産計画

	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2　高校時間割り編成問題のイメージ

	装置	要員	工具	8:30 9:00 9:30 10:00 … 5:30
タスク1	旋盤A	田中	R3.0	
タスク2	ドリル	佐藤	R2.0	
タスク3	鑽B	田中		
…				
タスクN	旋盤A	鈴木	R2.5	

図3　生産実行計画のイメージ

$x \neq y$; $C_2^{\text{東京, 千葉}}(x, y) = x \neq y$; ...) 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 [Selman], 各種グラフ問題, パズルなどがある.

2・2　時間割り編成問題と制約緩和問題 (CRP)

高校の時間割り編成問題 (図2) 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. 各授業の属性として, 担当の先生や対象クラス, 特殊教室など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, **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 (表1). 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.

2・3　生産計画問題と制約最適化問題 (COP)

一言で生産計画と言っても, 年間・月間予定を数表で計画する上位生産計画から, 1日の予定を線表で計画する生産実行計画まで, 多種多様である. 生産実行計画は, 各製品の各工程の作業など**タスク**が与えられ, そのタスクを実行するための**資源** (装置, 人員, 工具等) を時間軸上で割り付ける問題である (図3). 各タスクの開始時刻と利用資源が変数となり, 候補集合はそれぞれ1日の時刻の集合と各種資源の集合となる. 関連タスク (例えば前後工程の作業) の間には時間的な前後関係などの制約がある. 資源を時間軸上で同時に重複利用することはできない. その他, 納期, 能力, 作業切り換え時間など, さまざまな制約がある [野村 95].

ここで, まず, 問題となるのは, 生産コスト最小化・装置稼働率最大化など, 制約の他に最適化条件が与えられることである. そこで, **制約最適化問題** (COP) を導入する. COP は, CSP のモデルに加えて, 割り付けの良さ (または悪さ) を数値化する**目的関数**を持つ (表1). 問題は, 制約をすべて満足する割り付け (CSP の全解) の中で, 目的関数を最大化 (または最小化) するものを探索する**組合せ最適化問題**である.

CRP 同様, COP においても考慮制約が存在する.

	1週	2週	3週	4週	合計
Mate	100	150	80	120	450
…	…	…	…	…	…
Mobio	50	70	60	80	260

図4 上位生産計画のイメージ

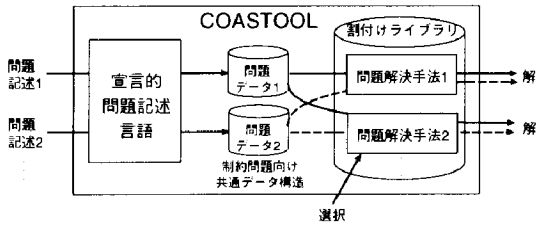

図5 COASTOOLアーキテクチャ

例えば, 納期や能力の制約は違反せざるを得ない場合が多い. この場合, 考慮制約の重要度に応じた違反点数を目的関数の中に組み込むことにより, COP として定式化可能となる. ここで, CRP は COP の一種であることに注意されたい. また, 違反点数は必ずしも定数ではなく, 納期遅れの程度など, 割り付け状況による関数となる場合が多い.

次に問題となるのは, 時刻の候補集合が時間軸上の無限集合となることである. この場合, CSP の解法は適用できない. また, 数表上の数値を決定する上位生産計画 (図 4) においても, 候補集合が無限集合であったり, 整数の有限集合ではあるが要素数が膨大なため無限集合として扱う場合が多い. 筆者は, スケジューリング問題を, 変数の型によって**点割り付け問題** (時間割りなど, 離散値問題), **量割り付け問題** (上位生産計画など, 連続値問題), **線割り付け問題** (生産実行計画など, 離散値・連続値混在型問題) に大別している.

3. スケジューリング問題のモデリング

筆者等は, 制約ベース計画シェル COASTOOL*5[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きた [野村 97, Shiina 95, 椎名 97]. 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, 制約問題は現実のスケジューリング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しないという課題がある. 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 5). 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 6), 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, 一つの問題に異なる解法を適用すること, また逆に, 異なる問題に一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.

```
(def-set 授業  数I1-1a 数I1-1b …) ;; 変数の集合を定義
(def-set 時限  月1 月2 … 土4)      ;; 候補集合を定義
(def-constraint 非常勤田中先生の都合          ;; 制約を定義
  :variables  ((v 田中授業))                  ;; 関与変数
  :condition  (not (is-a v 田中不在時間)) ;; 充足条件
  :penalty    10)                             ;; 違反点数
(def-problem 時間割り編成問題                 ;; CRP を定義
  :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
```

図6 COASTOOL/Lisp 問題記述例: 一部修正あり

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 1000 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 (ES) 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の実現において, 高速かつ高品質なデータ実装方法が重要である. COASTOOL/Lisp には, 問題の記述能力, メモリ容量, 速度性能, データベース (DB) や GUI との連携などの課

*5 COnstraint-based Assignment & Scheduling TOOL

```
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 ;};
   ...
}
```

図 7 COASTOOL/C++制約定義例: COASTOOL 提供クラス Constraint の派生クラスTeacherAbsence を定義. 先生と変数を属性に持つ. 規定の関数doCheck(), penalty() にて充足条件と違反点数を定義している.

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 7), 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, ある先生が三つの授業 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二つの授業の組合せすべてに対して, 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合せ的に爆発するという課題があり, 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. 処理の詳細は次の通り.

準備 あらかじめ先生時間割り表 (図 2(b)) のデータ構造を準備しておく.

割り付け 変数 (授業) に時限を割り付ける場合 COASTOOL は変数クラス Variable の規定関数 doTryValue() を呼び出す. この関数にユーザが定義したデーモンが起動され, 授業を先生時間割り表の対応する先生と時限の欄に登録する.

制約チェック さらに COASTOOL は, この変数に登録済みの仮想制約 C_0 を取得し, その関数 recheck() によりチェックする. ユーザが定義したrecheck() 関数は, 先生時間割り表の同じ欄に既に他の授業が登録済みであったか否かを調べる. 登録済みの場合は制約違反であり, この違反に対応するバイナリ制約 (C_1, C_2, C_3 のいずれか) を動的に生成する. また, 逆に, 違反でなくなったバイナリ制約は消去する.

もう一つの例として, 制約 6「先生の 1 日授業数の制限」を考える[*6]. 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, 違反する変数の組合せは ${}_{15}C_5 = 273$ 通りある. この大量の制約も, 一つの仮想制約と違反している制約の実体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[*7][Yoshikawa 96]. 仮想制約のメカニズムは複雑であり, その実現はけっして簡易ではない. 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. スケジューリング問題の解法

一般にスケジューリング問題は NP 完全であり, 残念ながら, ここに決定打となる解法を挙げることはできない. 経営工学 (OR) やエキスパートシステム (ES) の分野では, 個別問題向けの解法が多数開発されてきた. ここでは, 制約問題のモデルに基づく汎用的解法に重点を置き, その概要と動向を整理する.

4・1 制 約 伝 播

主な CSP の解法には, 無矛盾性 (consistency) 手法, バックトラック法, 局所探索法がある. **無矛盾性手法**は, **制約伝播 (constraint propagation)** による候補集合の絞り込みや新しい制約の導出など, 問題の等価変換を繰り返す. **制約論理言語**で用いられるが全解探索手法のため計算量が爆発する. このため制約伝播を他の手法と組み合わせて利用する場合が多い.

最も単純な制約伝播は**値伝播**である. 例えば, $x = y \vee z$ という制約があった場合, y や z の値の設定・変更・解除に応じて x の値を自動的に計算する. これは, 一種のデーモン (または**トリガ**) である. 値伝播は, 他の変数に従属な変数の計算を自動化するが, $x \Leftrightarrow y, z$ のような双方向的な計算の自動化は難しい.

*6 COASTOOL/Lisp では実装されていない.

*7 高校時間割り編成問題での比較.

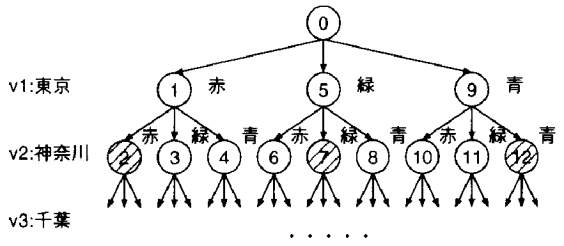

図 8　探索木

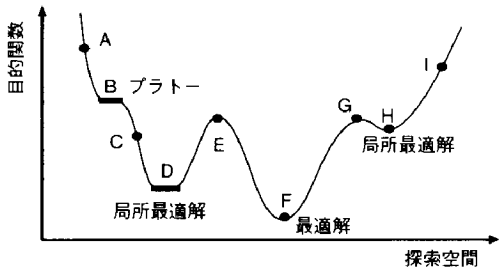

図 9　探索空間のイメージ図: 感覚的な理解のため, 探索空間を無理に 1 次元軸上に分布させている. 改良操作により曲線上を左右に少しずつ移動する.

双方向的な制約伝播にアーク無矛盾性 **(AC: arc consistency)** 手法 [Mackworth 92] がある*8. 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, 最終的に変化がなくなるまで伝播を繰り返す.

線割り付け問題で利用される**時性推論**は, 時間の前後関係に特化した制約伝播である. 複数の制約を同時に考慮する候補集合 (時刻の範囲) の絞り込み (path consistency) が実用的な速度で可能である.

4・2　バックトラック法

バックトラック法は, 基本的に, すべての組合せを一つずつ列挙し解かどうかを調べる手法である. 例えば, **2・1 節**の三色問題で, 変数 東京 は候補集合 { 赤, 緑, 青 } の中の 3 通りの値を取り得る. このそれぞれについて 神奈川 の取り得る 3 通りの値を考え, 順次, 最後の変数まで考える. ここで制約を調べれば全解を列挙することができる*9. 探索空間は, 図 8 のように**探索木**と呼ばれる木構造をなす.

バックトラック法も全解探索法であり計算量が爆発する. そこで, いかに早く最初の解に到達するかが課題になる. 図 8のノード 1 で, 変数 v_1 の値を 赤 にした際, $C_1^{1,2}(v_1, v_2) = v_1 \neq v_2$ を考えて v_2 の候補集合を { 緑, 青 } に絞り込めば*10, ノード 2 以下の部分木を調べる必要はない. このように, 制約伝播を用いて**枝刈り**を行うことにより探索木の走査を制限できる.

変数や値の選択順のヒューリスティックも重要である. 候補の数が少ない変数や関与する制約の数が多い変数を優先したり, 残りの変数への影響が小さい値を優先的に調べることにより, より早く最初の解に到達できる. また, ランダムな値の選択も有効である.

分枝限定 (branch & bound) 法は, バックトラック法と同様, 全解を調べながら最適解を求める COP の解法である. あるノード以下の解の目的関数が, 現在までの最適値よりも良くならないことが自明の場合には枝刈りを行う.

4・3　局所探索法

無矛盾性手法とバックトラック法は全解探索可能で計算時間が爆発するという課題がある. 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, 乱数等により**初期割り付け**を作成する. 通常, 初期割り付けは何らかの制約に違反している. 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れる点が応用に適している.

改良操作の内容や繰り返しの制御方法によりさまざまな方法がある. **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) に変更する山登り法である. 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.

図 9は探索空間のイメージ図である. 初期割り付けが E の場合, 山登り法を適用すれば最適解 (F) に辿り着く. しかし, 初期割り付けが C の場合, 局所最適解 (D) で止まってしまう. これは, 山登り法が改悪操作 (上向きの動き) を許さず, 丘 (E) を越えられないためである. そこで, ランダムな初期割り付けを与えてやり直す方法, ランダムに改悪操作も取り入れる手法, 局所最適解に陥ったときに改悪操作を用いて丘を越える**局**

*8　厳密にはバイナリ制約だけを持つ**バイナリ CSP** に対して定義される. 一般の CSP では Waltz's filtering. 任意の CSP はバイナリ CSP に変換可能だが実用性は疑問.
*9　生成テスト (generate & test) 法
*10　フォワードチェック (forward checking)

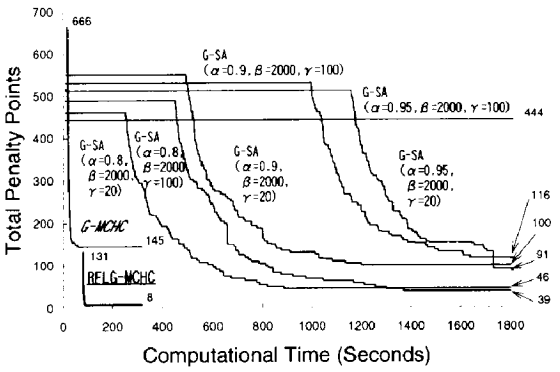

図10 高校時間割り編成の実験結果: G は Greedy 初期割り付け, SA でαは冷却比率, βは同温での繰り返し回数, γは初期温度. プロットは初期割り付け終了以降のみ.

所最適解脱出手法などが用いられる.

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り付け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.

タブーサーチ (tabu search)[Glover 93]は, 繰り返し改良時に最近の探索の履歴を**タブーリスト**に覚えておいて, これらを回避する変更操作を行う. 探索空間での後戻りやサイクルを避け, 長い繰り返しを要する局所最適解脱出にも効果がある. しかし, その実現には個々の問題に依存した作り込みが必要となる.

4・4 時間割り編成問題の解法

筆者等が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, 高品質の初期割り付け法である **RFLG 法**[*11][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 10).

RFLG 法は, 前述の AC 手法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 AC 手法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, 候補がなくなった変数は後回しにして **Greedy 初期割り付け法** により, 違反が最小の値を割り付ける. 制約伝播の利用により高品質の初期割り付けを作成可能である (図 10).

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 の改良は候補がな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 [金子 97a]. 局所最適解脱出手法は, 一度に同じクラスの二つの授業を玉突きのように動かす, 時間割り専用の方法である [金子 95]. これらの改良により, 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ない品質を得ることが可能となった.

4・5 生産計画問題の解法

生産実行計画問題は時間軸の扱いが課題であり, 汎用解法はほとんど実用化されていない. このため, OR/ES に見られるように個別問題向け解法の開発が進められてきた [宮下 95]. 時刻の候補集合が無限集合になるため, 例えば 10 分単位の時刻にして有限集合としたり, 時刻の代わりにタスクの順序を変数としたり[*12], 時刻だけを時性推論で処理する方法などが利用される. 最近は, 個別問題専用の繰り返し改良手法が主流である [Intell. Sched., Zweben 92]. 生産計画では, 計画遂行時の環境変化等に対して, 現行の計画を若干修正することにより対応する再計画が重要である. 繰り返し改良手法は再計画にも有効である.

上位生産計画問題は数値決定の課題があり, 汎用解法としては線形・非線形計画法が用いられる. しかし, そもそも制約問題としてモデル化することが困難な場合も多く, ES 的アプローチに頼るものも多い [Shiina 95, 椎名 97]. これは, 制約や目的関数を明記することが困難であったり複雑な場合である. むしろ専門家の計画立案の方法を明文化する方が容易な場合が多い. ES 的アプローチでは端数処理など数値決定のヒューリスティックが複雑になる.

4・6 主導権混在型問題解決

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[*13]. 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 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** [DITOPS, Yoshikawa 96] (図 11) が有効である.

*11 Really Full Lookahead Greedy 法

*12 基礎研究では一般的だが非実用的な場合が多い.

*13 CBR を用いた例もある [Miyashita 95].

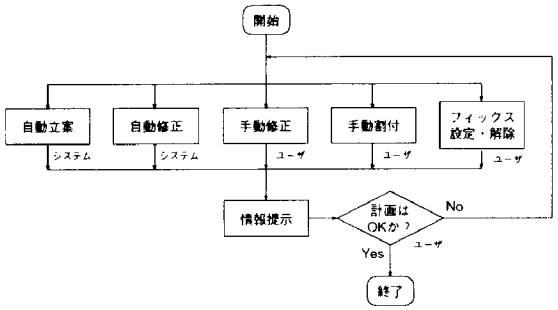

図 11　主導権混在型問題解決のフロー

例えば, 自動立案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, その結果を初期割り付け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往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, 割り付け不可能な理由を解析する説明機能などがある. これらを実装するうえで, COASTOOL のように制約問題 (特に制約) を宣言的なデータとして保持することが有効となる. また, システムとユーザが同一の問題データを共有することが重要である. 例えば, ユーザが制約や目的関数を修正した場合, 自動機能が即座に対応可能なことが望ましい.

5. 今後の課題と展望

以上, 汎用的問題解決に重点を置いて, 制約最適化技術のスケジューリング問題への応用について述べた. ここでは, 今後の課題と展望について述べる.

5・1 制約問題による定式化とモデリング

制約問題はスケジューリング問題のモデルとして有効である. また, 問題の定式化と解法とを切り離すことにより, 簡易なモデリングが可能となる. しかし, 制約の記述には研究者レベルの専門性が要求されるため, 簡易な制約記述方法の研究が重要である. 例えば, 例示プログラミングによる制約問題記述の研究が行われている [Minton 96a]. また, 効率的な制約の実装方法は応用面で極めて重要であり, 簡易な制約記述からの高度なコンパイル手法が望まれる.

現実の問題は, 例えば "いかに生産管理の品質を向上するか" であり, その回答はスケジューリングに限らない. 通常, 業務改革と併せたシステム導入が有効となる. このため, 制約問題のモデリングだけでなく, システム自身のプロトタイピング技術が重要である.

しかし, 残念ながら, この分野における知識獲得・学習の研究は, 解法の記述や探索制御に重点を置いており [宮下 96], 制約問題のモデリングはあまり研究されていないのが現状である.

5・2 制約問題の汎用的解法

時間割りなど, 一部の CRP に対して汎用的解法が成功している. しかし, 生産計画など, COP は個別解法に依存しているのが現状である. 局所探索法は有力であるが, 局所最適解脱出の問題や解発見の完備性の問題が課題となる. 近年, ヒューリスティックを用いて良さそうな解から順番に調べるバックトラック法が開発されている [Walsh 97]. これらは, 局所探索法同様, 常に近似解を保持しながら, 全解探索の完備性も備えており注目に値する. また, 主導権混在型問題解決は, モデリングと解法の課題に対して有効であり, 今後の発展が望まれる.

個別解法については, その解法の有効範囲を見極めることが重要であろう. 特に生産計画システムは, 多種大量のデータを利用し, 工場の組織構成, 生産対象, 生産設備, 管理業務形態, 管理戦略など, 実にさまざまな要素の影響を受ける. したがって, 個別のシステムを汎用化しパッケージ化すること, または, カスタマイズして転用することは容易ではない. また, 開発済みの生産計画システムを運用するためのデータベース (DB) メンテナンスには最低でも 2 ケ月を要するのが現状である. ここでは, 現実のニーズに基づくパッケージの要求の整理や DB 構築技術が有効になろう.

6. おわりに

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 [Galil 86]. 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. しかし, 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 OR/ES 的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.

しかし, Galil の言葉に対する回避策はないのであろうか? Minton は学習に基づく CSP 解法プログラム自動生成技術を開発し, Galil の言葉の反例を示した [Minton 96b]. また, 筆者等も COASTOOL により, 高校時間割りと全く同一の割り付けライブラリが, 特徴

を異にする大学時間割り編成に対しても有効であることを示した[金子 97b].

AI 研究の大きな目標の一つは汎用的問題解決である. 筆者は汎用の制約最適化技術が Galil の言葉への挑戦の糸口になればと考えている.

謝　　辞

最後に, 本稿執筆にあたりご指導頂いた日立製作所横田毅様, NEC 安倍直樹課長, 筆者の研究をご支援頂いた NEC 後藤敏所長, 阪田史郎所長, 島津秀雄部長, 渡辺正信マネージャ他の皆様に感謝いたします.

◇参　考　文　献◇

[DITOPS] *Ditops Home Page*, http://www.cs.cmu.edu/afs/cs/project/ozone/www/DITOPS/ditops.html.

[Galil 86] Z. Galil, "Efficient algorithms for finding maximum matchings in graphs," *Computing Surveys*, 18(1), 1986.

[Glover 93] F. Glover, E. Taillard, and D. de Werra, "A User's Guide to Tabu Search," *Anals of Operations Research*, 41:3–28, 1993.

[i2] *i2 technologies Home Page*, http://www.i2.com.

[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).

[Intell. Sched.] M. Zweben and M. S. Fox (eds.), *Intelligent Scheduling*, Morgan Kaufmann, 1994.

[JPL] *JPL Home Page*, http://www-aig.jpl.nasa.gov/index.htm

[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, 1995, 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.

[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.

[Miyashita 95] K. Miyashita and K. Sycara, "CABINS: A Framework of Knowledge Acquisition and Iterative Revision for Schedule Improvement and Reactive Repair," *Artif. Intell.*, 76:377–426, 1995.

[宮下 95] 宮下, "スケジューリング問題へのアプローチ (1) - 人工知能における課題-," 人工知能学会誌, 10(6):880–887, 1995.

[宮下 96] 宮下, "スケジューリング問題へのアプローチ (2) - 知識の内容とその獲得-," 人工知能学会誌, 11(1):41–49, 1996.

[野村 95] 野村, 岩本, 山之内, 渡辺, "プロセス産業を対象としたスケジューリングアーキテクチャ," 経営システム, 5(1):27–34, 1995.

[野村 97] 野村, 岩本, "数量組合せ同時最適化手法の提案と適用評価" 生産スケジューリングシンポジウム'97, pp. 79–84, 1997.

[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.

[PeopleSoft] *People Soft Home Page*, http://www.peoplesoft.com/products_and_services/enterprise_solutions/redpeppr.htm.

[Selman] *Bart Selman's Home Page*, http://portal.research.bell-labs.com/orgs/ssr/people/selman.

[Shiina 95] S. Shiina, M. Yoshikawa, T. Yamanouchi, and M. Watanabe, "PC Master Production Scheduling System with Task/Resource/Quantity Selection Heuristics," *Proc. 7th Int'l Conf. on Tools with AI*, pp. 79–86, 1995.

[椎名 97] 椎名, 吉川, "逆 MRP - 部品納入予定を考慮した上位生産計画方式," 生産スケジューリングシンポジウム'97, 1997.

[Walsh 97] T. Walsh, "Depth-bounded Discrepancy Search," *Proc. 15th Int'l Joint Conf. on AI*, 2:1,388–1,393, 1997.

[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 回人工知能学会全大, No. 14-7, 1993.

[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.

[Zweben 92] M. Zweben, E. Davis, B. Daun, E. Drascher, M. Deale, and M. Eskey, "Learning to improve constraint-based scheduling," *Artif. Intell.*, 58:271–296, 1992.

著　者　紹　介

吉川　昌澄(正会員)

1984 年名古屋大学理学部物理学科卒業. 1986 年同大学工学部情報工学科修士課程修了. 同年 NEC 入社. 1990 年まで Common LISP 言語処理系の開発に従事. 1991 年より制約最適化, スケジューリング問題の研究開発に従事. 1995～1996 年 USC/ISI 客員研究員として Steven Minton 博士に師事. 現在, NEC C&C メディア研究所システムソリューションテクノロジーグループ主任. 情報処理学会, AAAI 各会員. 人工知能学会知識ベースシステム研究会 (SIG-KBS) 幹事. 時間割り自動編成の実践と理論国際会議 (PATAT) 運営委員.

〈yosikawa@ccm.cl.nec.co.jp〉