数理解析研究所講究録
1015巻 1997年 124-136

汎用アルゴリズムとしての CSP(制約充足問題)に対するタブー探索アプローチ

野々部 宏司　　茨木 俊秀
Koji NONOBE　　Toshihide IBARAKI

京都大学大学院工学研究科数理工学専攻
〒606-01 京都市左京区吉田本町
Department of Applied Mathematics and Physics,
Graduate School of Engineering, Kyoto University
Kyoto 606-01, Japan
{nonobe, ibaraki}@kuamp.kyoto-u.ac.jp

Abstract

Many application problems encountered in the real world are combinatorial (optimization) problems, e.g., scheduling, allocating problem, assignment problem, and timetabling. Constraint satisfaction problem (CSP) can naturally formulate these problems. Therefore, an efficient CSP algorithm can solve all such problems, and can be used as a general problem solver. In this paper, we develop a tabu search based CSP algorithm, incorporating some elaborations, such as an automatic control mechanism for the tabu tenure, modification of the penalty function to handle objective functions, and enlargement of the neighborhood by using shift and swap operations, in addition to the basic building blocks of tabu search. We will demonstrate the effectiveness of our CSP algorithm by computational experiment for graph coloring, timetabling, and nurse scheduling problems.

Key Words: combinatorial problem, general problem solver, CSP, meta-heuristics, tabu search.

1 はじめに

現実社会に現れる問題には, 配置問題や輸送問題, スケジューリング問題のような組合せ問題 (組合せ最適化問題) がその中心となっていることが少なくない. しかし, その多くはNP困難であり, 厳密解を効率良く求めるアルゴリズムは存在しないことが理論的にも強く示唆され, 大規模な問題を厳密に解くことは事実上不可能であると考えられる. このことから, 必然的に近似解法, 発見的手法といった研究が重要となり, 様々な組合せ問題に対して数多くの手法が提案されてきた. しかし, それらの多くは, 与えられた個々の問題に対して, その問題構造をうまく利用して解を求めるという専用アルゴリズムであった. 専用アルゴリズムは非常に有効ではあるが, アルゴリズムの構築には多くの手間と時間を要するものであり, (特に現場の人達にとって) 手軽に試みることは容易ではない場合が多い. そこで本研究では, 多くの組合せ問題を一括して扱うことのできる汎用近似アルゴリズムの構築を目指す.

汎用アルゴリズムは, 多くの組合せ問題を一括して扱い得るだけでなく, 与えられた問題に対して, その問題構造に関する深い知識をあまり必要とせずに, しかも良質の近似解を実用的な時間で求め得るものでなくてはならない. そこで本研究では, 高い定式化能力を持つ制約充足問題 (Constraint Satisfaction Problem, CSP) に着目し, まず与えられた問題を一旦CSPに定式化した後, CSPアルゴリズムを用いて解くことを考える. CSPは主に人工知能の分野で研究されており [4, 7, 8, 14, 16, 18], 比較的以前から研究されている拘束伝播, backtracking法など厳密解を求めることに主眼をおいた手法に加えて, 近年では, 局所探索法 (local search) (反復改善法 (iterative improvement), 山登り法 (hill climbing) とも呼ばれる) の一種であるMCHC法 (minimum conflict hill climing)

[14], メタ戦略 (meta-heuristics) の一つである遺伝的アルゴリズム (genetic algorithm, GA) や, ニューラルネットワーク (neural network) を用いた解法などの近似解法 [2, 3, 12] も研究されている. 本研究では, 多くの組合せ問題に対して比較的容易に適用でき, 成功を収めているメタ戦略に注目し, その中でも特に, 非常に柔軟な枠組みを持つタブー探索 (tabu search) [9, 10] が汎用アルゴリズムとしてのCSP アルゴリズムの枠組みに適していると考え, CSP に対するタブー探索の適用を行った. その際, パラメータの自動調節等, 幾つかの工夫を加えることで, 探索能力の向上, 及びユーザの手間の削減を図っている. 計算実験の対象には, グラフの彩色問題, 一般化割当て問題, 集合カバー問題, ブロック計画, 時間割問題, スケジューリング問題等を扱ってきた [17]. 本論文では, その中の幾つかについてやや詳しく述べる.

2 制約充足問題

CSP は, n 個の変数 X_i $(i = 1, 2, \cdots, n)$ とそれぞれに対応する有限離散領域 D_i, 及び m 個の制約 $C_l(X_{l_1}, X_{l_2}, \cdots, X_{l_{t_l}})$ $(l = 1, 2, \cdots, m)$ で定義され, 全ての制約を満たすように, 各変数 X_iに値 $j \in D_i$を割当てる問題である [4, 14, 16, 18]. ここで, 各制約 C_lは変数 $X_{l_1}, X_{l_2}, \cdots, X_{l_{t_l}}$ に対する t_l-項制約であり, それらの変数の値域の直積 $D_{l_1} \times D_{l_2} \cdots \times D_{l_{t_l}}$の部分集合である. 全ての制約を満たすような値の割当てを実行可能解と呼び, そのような解 (存在しない場合はできるだけ多くの制約を満たす解) を 1 つ見つけることが目的である.

ここで, 変数 X_iとその値 j $(\in D_i)$ の組それぞれに対して, 値変数 x_{ij}を

$$x_{ij} = \begin{cases} 1, & \text{変数 } X_i\text{が値 } j\text{をとる,} \\ 0, & \text{その他,} \end{cases}$$

と定義し, 割当てを$\sum_{i=1}^{n} |D_i|$ 次元の 0-1 ベクトル $\boldsymbol{x} = (x_{ij} \mid i = 1, 2, \cdots, n, j \in D_i)$ で表す. このとき,

$$\sum_{j \in D_i} x_{ij} = 1, \quad i = 1, 2, \cdots, n, \tag{1}$$

が成立すれば, 全ての変数 X_iに対して値が 1 つ割当てられていることになる. 以下では, 簡単化のため $D_1 = D_2 = \cdots = D_n = D$ とする. 制約 (1) は割当て $\boldsymbol{x}$ がCSP の解となるための必要条件であり, 本研究におけるタブー探索では, 制約 (1) を満たす範囲内で探索を行う.

また, 制約 C_lをどのように記述するかは一意ではなく, 方程式, 不等式, 論理式, 値の組の集合, あるいは判定のアルゴリズムを直接準備する等, 問題に応じて適当な表現方法を用いることができる. このように制約の記述方法を問題に合わせて選べることは, 問題の定式化をコンパクトに行えることにつながり, 制約の処理の高速化に役立つ. これは, 整数計画法等にはない, CSP アルゴリズムの特長の 1 つであると言える. 現在のところ, 標準的な表現方法として, 線形等式, 線形不等式に加え, 変数集合 $V' \subseteq V$内の変数は全て異なる値を取るという not-equal-values 制約を組み込んでいる.

3 タブー探索の基本的枠組み

タブー探索中, 式 (1) を満たす割当て全てから成る探索空間 $\mathcal{X}$を考え, $\boldsymbol{x} \in \mathcal{X}$の近傍 $N(\boldsymbol{x})$ を, ある 1 つの変数 X_i の値 jを他の値 $j' \in D$に変えることによってできる割当ての全てとする (以下, これを shift 近傍と呼ぶ). 更に, 各制約 C_lに対して, それが満たされるならば 0, 満たされないならば正の値をとるようなペナルティ関数 $p_l(\boldsymbol{x})$ を定義する. 例えば, 線形不等式 $\sum_{i,j} a_{ij} x_{ij} \leq b$ に対しては $\max\{\sum_{i,j} a_{ij} x_{ij} - b, 0\}$, 変数集合 V'に対する not-equal-values

制約に対しては, $|V'| - |\{j \in D \mid \exists X_i \in V', X_i = j\}|$ と定義すれば良い. これらのペナルティ関数を用いて, 全体としてのペナルティ関数 $p(\boldsymbol{x}) = \sum_l p_l(\boldsymbol{x})$ を考えることで, 本来決定問題である CSP を最小化問題

$$\begin{array}{ll} \text{minimize} & p(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in \mathcal{X}, \end{array} \tag{2}$$

として扱うことができる. すなわち, ペナルティ関数値が 0 であるような解 $\boldsymbol{x} \in \mathcal{X}$が存在することと, CSP が実行可能解を持つことは同値である.

また, タブー探索では, 局所探索において改悪解への移動を許可しているので, 通常, 解の移動の属性をタブーリストに記憶しておき, ある期間 t (tabu tenure と呼ばれる) その逆向きの移動を禁止するという方法がとられる. タブーリストに記憶しておく属性としては, 値が 1 から 0 に変わった値変数 x_{ij} と値が割当て直された変数 X_i との 2 つが考えられ, どちらが適しているかは問題に依存する. しかし, 一般に前者の方が解の移動を忠実に表現していると言え, その分タブーリストによる制限が緩く, 後者は変数 X_i に関わる移動を全て禁止してしまうということから, 探索が制限され易いといえる. このことから, 比較的規模が小さい (探索空間が狭い) 問題に対しては前者が, 大規模な問題に対しては後者が適していると考えられる.

また, タブー探索の基本的要素として, aspiration criteria や長期メモリによる多様化もよく用いられる. aspiration criteria は, タブーリストにより禁止されている移動でも, ある基準を満たせば許可されるというものであり, その基準として, "これまでの暫定解を更新する場合" とすることが多い. 後述するように, 本アルゴリズムでは, この基準を若干緩め, さらにその結果を tabu tenure の自動調節に用いている. 長期メモリは, 配列 LTM を用意し, $LTM(x_{ij})$ に値変数 x_{ij} が 0 から 1 に変わった回数 (変数 X_i の値を j に割当て直した回数) を記憶しておくことで実現している. その結果, 各反復の近傍探索において解の評価を行なう際に, LTM をペナルティ項として加えたり, タブー探索を何度か反復して行う場合, 次の探索の初期解の生成に LTM を反映させたりすることで多様化を図ることが可能となる. しかし, これらの手法は何らかのパラメータを伴うことが多く, その調整が問題となってくるため, 現段階では, 近傍探索の際の tie break にのみ用いる. すなわち, 近傍内に最良解が複数存在した場合, LTM の値が小さい方を選択することにする.

この他, 以下の手法を組み込んでいる.

・欲張り法による初期解の生成

探索の初期解の生成のための最も簡単な方法はランダムに解を生成することであるが, そのような解は一般に質が悪く, 容易に改善することができる. しかし, タブー探索は (改善解が見つかった時点で近傍探索を打ち切る first 改善ではなく) 基本的に近傍内に含まれる解全てを調べる best 改善であるため, 質の悪い解から始めると, 探索に多大の時間を要し, 効果的であるとは言えない. そこで, 少ない計算量で, ある程度良質の解を得ることができる欲張り法 (greedy method) により初期解を生成する. 具体的には, 何も値を割当てていない状態から始めて, 関係する制約が多い変数 (すなわち影響力の大きい変数) から順に, できるだけペナルティ $p(\boldsymbol{x})$ が増加しないように値を割当てていく. 但し, この方法により複数個の解を生成する場合は, 多様性に欠ける恐れがあるため, 先に述べた長期メモリを用いる方法や, GRASP (greedy randomized adaptive search procedure) [5, 13] のように, 欲張り法にランダム性を加える方法などが考えられる.

・近傍の削減

タブー探索においては, 各反復において解を更新する際, 近傍内の解を全て調べるのが基本であるが, (特に大規模な問題例に対しては) 多くの計算時間を費やしてしまう. そこで, 本アルゴリズムでは, 現在満たされていない制約に関係している変数の値を変更することによって得られる解のみを次の解の候補とし, 近傍の削減を行っている.

4 Tabu tenure の自動調節

タブー探索では, プログラムパラメータである tabu tenure t の値が探索能力に大きな影響を与えるため, その調整に多大な時間を要することが報告されている. そこで, t を自動調節するメカニズムを組み込むことで, この調整手間を削減することを考える. 但し, 汎用アルゴリズムとして用いるには, 個々の問題構造に依存した手法ではなく, 探索状況に応じて t を自動的に調節することが望ましい. このような方法の 1 つとして, 過去に探索した解を (直接的に, あるいはハッシュ関数等を用いて近似的に) 記憶しておき, 解のサイクリングが検出されると t を増加させ, 逆にサイクリングがある程度の期間生じなければ減少させるという, Reactive tabu search [1] が知られており, 幾つかの組合せ問題に対してその有用性が示されている. しかし, この方法では, 過去に探索した解を再び探索しない限りサイクリングとは見なされないため, 探索空間が広い大規模な問題例において, サイクリングではないものの, 探索空間のある一部を克明に調べすぎるという恐れがある. そこで本研究では, Reactive tabu search とは異なる方法を提案し, CSP アルゴリズムに組み込んだ. 以下, タブーリストには値を変更した変数 X_i を入れておくものとして, その概略を述べる.

4.1 Tabu tenure の増加

tabu tenure t は, 現在の探索が解空間のある一部分に限定されており, 多様化が必要と判定されると $t := t + 1$ に増加される. その判断は次のように行われる. 第 k 反復において, 解 $\boldsymbol{x}^{(k-1)}$ から $\boldsymbol{x}^{(k)}$ に移動する際に値を割当て直した変数を $X^{(k)}$ とし, 第 k 反復までに値を割当て直した変数集合を $A^{(k)}$ とする. すなわち, $A^{(0)} := \phi, A^{(k)} := A^{(k-1)} \cup \{X^{(k)}\}$ である. このとき, ある反復 k において, 変数 $X^{(k)}$ が k 以前に第 $k'(< k)$ 反復で値が割当て直されており, しかも $A^{(k')} = A^{(k)}$ であったとすると, この間の探索においては, 同じ変数集合 $A^{(k)}$ 内の値を変化させているだけであり, 探索が偏っていると考えられる. そこで, このような場合には多様化が必要であると判断し, t を 1 増加させる. 但し, このままでは $A^{(k)}$ が単調増加であるため, ある程度反復が進むと毎回 t が増加してしまうという結果になるので, それを防ぐため, 暫定値が更新されたり, 多様化が行われたと判断されたときには, $A^{(k)} := \phi$ として過去の情報を消去する.

多様化が行われたと判断する基準としては, t の増加 (多様化) に伴って初めて値が割当て直された変数で, しかもその際, 解の移動が改悪であった (ペナルティ関数値が増えた) ような変数 X' に着目し, 変数 X' が, 再びタブーリストから出た直後に値を割当て直されなかった場合, 多様化が行われたと判断する. すなわち, 初めに変数 X' の値が割当て直されたときは改悪であったので, その直後は変数 X' の値を元に戻せばペナルティ関数値を減らすことができる. しかし, しばらくの間タブーリストの働きによりその移動は禁止されており, 変数 X' がタブーリストから出たにもかかわらず値を元に戻されなかったということから, その間に解が充分変化し, 多様化が行われたと判断するわけである. 従って, 逆に変数 X' の値が割当て直された場合には, タブーリストの長さが十分でないと判断し, t を 1 増加させる.

4.2 Tabu tenure の減少

次に tabu tenure t の減少法について考える. 前章で述べた通り, タブー探索では, ある解の移動がタブーリスト T により禁止されていたとしても, aspiration criteria を満たせば, その移動を許すという手法が通常用いられる. ここでは, この基準をサイクリングが起こらない程度にまで緩め, 探索中, タブーリストにより禁止されている移動が, aspiration criteria を満たすことによって実行された場合には, $t := t - 1$ の減少を実行する.

aspiration criteria には, 前章で述べた標準的な aspiration criteria に加えて, 以下のルールを用いる. 現在第 k

反復であるとする. ある変数 $X^{(k')} \in T$ $(k' < k)$ が第 k' 反復において T に入ったときの解の移動が改善であり, しかも, 今 $X^{(k')}$ の値を割当て直すことで, 第 k' 反復と比べてもペナルティ関数値を更に小さくすることができるならば, 変数 $X^{(k')}$ の値の変更がサイクリングを起こすことはないと判断して, その変更を許可する.

4.3 計算実験

tabu tenure の自動調節機能の効果を見るために, グラフの彩色問題に対して計算実験を行った. 計算実験は全て Sun Ultra 2 Model 2200 (200MHz) 上で行い, 言語は C 言語を用いている. グラフの彩色問題は, 与えられた無向グラフ $G(V, E)$ に対し, 隣接する節点は異なる色となるように全ての節点に色を塗る問題である (そのような彩色を実行可能であると呼ぶ). ここでは, グラフの彩色問題を CSP の枠組みで扱うため, 決定問題 (すなわち, 無向グラフ $G(V, E)$ を k色で彩色可能かどうかを決定する問題) として考える. すなわち, 節点 $v_i \in V$ を変数 X_i に対応させ, 領域 D を $D = \{1, 2, \cdots, k\}$ とし, 枝 $e_l = (v_{i_1}, v_{i_2}) \in E$ に対して, 制約 C_l: "変数 X_{i_1} と X_{i_2} は異なる値をとる" を考える. 制約 C_l は, k個の線形不等式を用いて,

$$x_{i_1 j} + x_{i_2 j} \leq 1, \qquad j = 1, 2, \cdots, k, \tag{3}$$

と記述することもできるが, 先に述べた not-equal-values 制約を用いることによって, 計算時間の削減を図っている (以下の問題例に対しては, 約 7〜30 倍速く計算できることが計算実験により確認された).

計算実験には, DIMACS のベンチマーク問題の中から, Leighton graph と呼ばれるグラフを用いた. これらのグラフに対しては, 彩色可能な最小の色数 k が知られている. tabu tenure t を 10, 20, 30 に固定した場合と自動調節した場合それぞれについて, 初期解を変えて 10 回ずつ探索を行った. 各探索は, 実行可能な彩色が得られるか, 計算時間が 300 秒に達するまで行った. 実行可能な彩色が得られた場合, その探索は成功であり, そうでない場合, 失敗であると呼ぶ. 表 1 に計算結果を示す. $|V|$, $|E|$, k はそれぞれ節点数, 枝数, 最小彩色数であり, 10 回の探索が全て成功であった場合には上段に平均 CPU 時間 (秒) を, 下段に平均反復回数 (回) を示し, 1 回でも探索が失敗した場合には上段に成功した探索回数 (10 回中) を, 下段に平均暫定値 (暫定解において満足されていない枝の本数の平均) を示している. t を自動調節した場合, 常に最良の結果が得られているわけではないが, t の適正値が問題例によって異なるにもかかわらず, 全ての問題例に対して比較的良い挙動を示している. 自動調節によって, 前もって t の調整をする必要がないことを考慮すれば, 非常に有用であると言える.

また, ランダムグラフに対しても実験を行い, 同様の結果を得ている. 更に, 同じ問題例に対する他の計算結果 [6] と比較した結果, 本アルゴリズムがグラフの彩色問題に対して, 専用アルゴリズムと同程度の性能を有することが確認された. 詳しくは [17] を参照されたい.

5 目的関数の導入

CSP は, 実行可能性のみを判断する決定問題であるが, 現実の場面において, 目的関数を伴う最適化問題を扱わねばならない場合も多い. 本章では, 目的関数を CSP の枠組みで扱うことを考える. 以下, 簡単化のため, 目的関数は整数係数 c_{ij} を用いて

$$f(\boldsymbol{x}) = \sum_{i,j} c_{ij} x_{ij}, \tag{4}$$

と表せ, 最小化問題であると仮定する.

目的関数 $f(\boldsymbol{x})$ を導入する直接的な方法は, 問題 (2) を

表 1: グラフの彩色問題に対する計算結果.

問題例	$\|V\|$	$\|E\|$	k	t : 固定			t : 自動調節	
				$t=10$	$t=20$	$t=30$		t の平均値
le450_5a.col	450	5714	5	3.8 6988.4	1.4 1928.4	2.9 5791.8	1.9 2896.9	13.2
le450_5b.col	450	5734	5	9/10 (6.1)	3.3 7499.6	5.3 11925.8	2.6 4686.2	16.7
le450_5c.col	450	9803	5	7/10 (10.1)	0.8 921.6	0.8 1097.7	0.9 1264.5	14.2
le450_5d.col	450	9757	5	1.5 2040.4	0.6 648.2	0.6 697.2	1.3 1815.3	32.5
le450_15a.col	450	8168	15	0/10 (5.1)	0/10 (8.8)	0/10 (11.9)	0/10 (6.3)	14.7
le450_15b.col	450	8169	15	0/10 (4.3)	0/10 (8.1)	0/10 (11.8)	0/10 (5.8)	14.3
le450_15c.col	450	16680	15	0/10 (169.3)	110.4 56430.8	48.8 32608.6	81.6 38275.2	20.8
le450_15d.col	450	16750	15	0/10 (176.1)	8/10 (0.2)	78.5 65036.5	106.1 70516.1	21.1
le450_25a.col	450	8260	25	1.4 1669.2	2.2 2872.2	20.0 19186.0	0.4 314.8	4.6
le450_25b.col	450	8263	25	0.4 258.4	0.6 237.9	1.4 559.0	0.2 39.9	2.1
le450_25c.col	450	17343	25	0/10 (17.0)	0/10 (20.0)	0/10 (24.9)	0/10 (17.7)	15.4
le450_25d.col	450	17425	25	0/10 (15.8)	0/10 (20.2)	0/10 (24.1)	0/10 (16.8)	15.6

$$\begin{aligned} &\text{minimize} \quad q(\boldsymbol{x}) = w_o f(\boldsymbol{x}) + p(\boldsymbol{x}) \\ &\text{subject to} \quad \boldsymbol{x} \in \mathcal{X}, \end{aligned} \tag{5}$$

と変更することである. ここで, $w_o > 0$ は $f(\boldsymbol{x})$ に対する重みであり, プログラムパラメータである. w_oが大きいと, 目的関数値は小さくなり易いが, 実行可能解が得にくくなる. 逆に w_o が小さいと, 解は実行可能解になり易くなるが, 目的関数を小さくすることが困難となる. このように, 計算性能が重み w_oに強く依存してしまうという欠点がある.

目的関数を導入するもう 1 つの方法として, zを $f(\boldsymbol{x})$ の当面の目標値として,

$$f(\boldsymbol{x}) \leq z, \tag{6}$$

を制約として追加する, いわゆる目的関数固定法がある. 例えば暫定値が得られている場合, zを ($f(\boldsymbol{x})$ の暫定値)-1 とし, 暫定値が更新される度に zを減少させていくという方法が考えられる. この方法は,

$$\begin{aligned} &\text{minimize} \quad q(\boldsymbol{x}) = w_o \max(f(\boldsymbol{x}) - z, 0) + p(\boldsymbol{x}) \\ &\text{subject to} \quad \boldsymbol{x} \in \mathcal{X}, \end{aligned} \tag{7}$$

と書くことができる. $w_o > 0$ は, (5) と同様, 制約 (6) に対する重みである. この方法は (5) ほど w_oの値に強く依存することはないが, zより小さい目的関数値を持つ解に対しては, 全て等しく $q(\boldsymbol{x}) = p(\boldsymbol{x})$ であり, $f(\boldsymbol{x})$ を zより小さくしようとする作用はない. そのため, 一般に (5) と比べて良質の解を求める能力は劣る.

そこで, 上の 2 つの折衷案として

$$\begin{aligned} &\text{minimize} \quad q(\boldsymbol{x}) = w_o \{\max(f(\boldsymbol{x}) - z, 0) + \theta \min(f(\boldsymbol{x}) - z, 0)\} + p(\boldsymbol{x}) \\ &\text{subject to} \quad \boldsymbol{x} \in \mathcal{X}, \end{aligned} \tag{8}$$

を考える. ここで, $0 \leq \theta \leq 1$ はプログラムパラメータであり, $\theta = 1$ とすれば (5) と, $\theta = 0$ とすれば (7) とそれぞれ等価になる. 例えば $\theta = 0.5$ とした場合, (5) において w_oが適正値に設定された場合にはやや劣るものの, (7) よりは良質の解を求める能力があるという意味で優れており, 重み w_oに強く依存してしまうことがないという意味で (5) よりは安全であると言える. このことから, $\theta = 0.5$ とすることが推奨される.

先に述べた通り, w_oの値は探索中の解の実行可能性に大きな影響を持っている. このことを利用して, 探索中, 訪れた解が実行不可能である割合がほぼ $[LB, UB]$ の範囲に入るように w_oを調整する. 具体的には, 最近 100 回の反復において, 解が実行不可能である割合が LB以下 (UB以上) であるならば w_oを σ 倍する (σ で割る). 但し, $0 < LB < UB < 1$, $\sigma > 1$ であり, LB, UB, σはそれぞれプログラムパラメータである. この手法では, 1 つのパラメータ w_oを調節するために, 3 つのパラメータが必要となる. しかし, w_oの適正値は, 同じ問題でも問題例によって大きく異なるのに対して, LB, UB, σの適正値は問題例にさほど依存しないという利点がある. 一般的に, 実行可能解を得ることが困難な問題に対しては, LB, UBの値を大きく (例えば $[LB, UB] = [0.6, 0.8]$ などに) 設定すると効果的である.

6 swap 近傍

問題によっては, 2章で定義した shift 近傍だけでは効果的な探索ができないことも多い. 例えば,

$$\sum_i x_{ij} = a_j \text{ (定数)}, \qquad j \in D, \tag{9}$$

という制約を含む問題においては, 制約 (9) を満たす解 $\boldsymbol{x}$ の shift 近傍内に制約 (9) を満たす解は存在せず, 解の移動を行う度に, 制約 (9) を一旦破らなくてはならない. そこで, 本章では swap 近傍を導入することで近傍の拡張を行う. swap 近傍とは, ある 2 つの変数 X_{i_1}, X_{i_2}にそれぞれ値 j_1, j_2 ($j_1 \neq j_2$) が割当てられている場合, 変数 X_{i_1}の値を j_1から j_2に, 変数 X_{i_2}の値を j_2から j_1に変更することによって得られる解の集合である. この近傍を用いることにより, 制約 (9) を必ずしも破ることなく探索を行うことが可能となり, より効果的な探索ができる.

しかし, swap 近傍の大きさは $O(|V|^2)$ であり, 特に変数の数が多い問題例に対しては, 計算時間を減らすために何らかの工夫をしなくてはならない. ここでは, (i) まず shift 近傍内の解を調べ, その中に改善解が存在しない場合のみ, swap 近傍も調べる, (ii) swap 近傍に対しては first 改善とする. すなわち, 改善解が見つかった時点で近傍探索を打ち切る, という方法を採用している.

swap 近傍の効果を見るために, 大学における時間割問題 [15] に対する計算実験を行った. この問題は, 30 の講義を, いつ, どの教室で行うかを決定する問題である. 時間数は 10 ($T_1 \sim T_{10}$), 教室数は 3 ($R_1 \sim R_3$) であり, 各教室には収容人数が定められている. 先生は 13 人であり, 予めどの先生がどの講義を行うかは決定されており, 各先生には講義を行いたくない時間が幾つかある. また, 60 人の学生は, それぞれ 30 の講義から履修したい講義を 8〜10 選択しており, 履修希望人数の多い講義は, 収容人数の少ない教室で行うことはできない.

この問題は, 変数の数 $n = 30$ (講義数), 領域の大きさ $d = 10$ (時間数) $\times$ 3 (教室数) とすれば容易に CSP に定式化できる. また, 以下の 5 つが制約であり, 774 の線形不等式制約で記述できる.

制約 1. 同じ時間に同じ教室で異なる講義を行わない.

制約 2. 同じ先生の講義は同じ時間に行わない.

制約 3. 同じ学生の希望する講義は同じ時間に行わない.

制約 4. 先生の希望しない時間に講義を行わない.

制約 5. 収容人数の足りない教室で講義を行わない.

この問題は制約が厳しく, 全ての制約を満たすような解は存在しない. そこで, 全制約を絶対制約 (hard constraint) と考慮制約 (soft constraint) に分ける. 絶対制約は, 解が実際に意味のある時間割となるために必ず満たされなくてはならない制約で, 制約 1., 2., 5. がそれにあたる. 残る制約 3., 4. は考慮制約であり, 満たされることが望ましいが, 必ずしも満たされなくても良い. これにより, 絶対制約を全て満たしつつ, 考慮制約をできるだけ多く満たすことが目標となる. これを実現するために, 制約 C_l に対するペナルティ関数 $p_l(\boldsymbol{x})$ に正の重み w_l をかけ $p(\boldsymbol{x}) = \sum_l w_l p_l(\boldsymbol{x})$ とし, 絶対制約に対しては w_l を十分大きく (ここでは 100), 考慮制約に対しては小さく (ここでは 1) 設定する.

この問題例に対して, shift 近傍のみ, shift 近傍+swap 近傍のそれぞれについて 10 回ずつ探索を行った. 各探索は 60 秒間行った. 表 2に, ペナルティの平均, 及び 10 回の探索で得られた最良解について, 履修することができない講義数がそれぞれ 0, 1, 2, 3, 4 以上である学生の数, 先生の希望に対する違反数, 及びペナルティの合計値が示されている. この結果から, swap 近傍を導入することにより探索能力が向上していることが確認できる.

表 2: 大学の時間割問題に対する計算実験.

近傍	ペナルティの平均	10 回の探索における最良解						
		履修できない講義数とその学生数					先生の希望に対する違反数	ペナルティ
		0	1	2	3	4 以上		
shift 近傍	88.5	6	27	23	4	0	2	87
shift 近傍 + swap 近傍	84.8	7	29	19	5	0	2	84

7 現実問題に対する計算結果

本研究は汎用アルゴリズムの実現を目指すものであり, 実社会に現れる多様な問題に対する有用性を示すことが望まれる. そこで, 現実問題として高等学校の時間割問題, 及び看護婦スケジューリング問題に対する計算実験の結果を示す.

7.1 高等学校の時間割問題

ここで扱う問題例は東京郊外の高等学校における時間割問題であり, [19] で扱われている. この高校には 30 クラス, 60 人の先生, そして 3 つの特別演習室があり, 780 の授業と 13 の会議を何曜日の何時限目 (月曜から金曜は 6 限まで, 土曜は 4 限まで) に, 行うかを決定しなくてはならない. ここで, 各授業には科目, 先生, クラス, 教室が割当てられており, 各会議にはどの先生が出席するかが定められている. この問題を CSP に定式化するためには, $n = 780 + 13 = 793$ 個の変数と, 領域 $D = \{$ 月 1, 月 2, $\cdots$, 土 4$\}$ ($|D| = 34$) を導入すればよい. これにより, 値変数 x_{ij} の数は $793 \times 34 = 26962$ となる. 制約は, 時間割問題における典型的なものから, この問題例に特有なものまで, 数多くある. その幾つかを以下に述べる.

- 各クラスは, 各時限において高々1 つの授業しか行えない.
- 各先生は, 各時限において高々1 つの授業, または会議しか行えない.
- 各特別演習室においては, 定められた数までの授業しか同時に行うことはできない.
- 各先生が 1 日に行うことのできる授業, または会議には上限がある.
- 幾つかの授業は 2 駒連続して行わなくてはならない (芸術など).
- 各クラスにおいて, 同じ科目を 1 日に 2 駒以上行ってはならない (2 駒連続の授業を除く).
- 1 週間に 2 駒しかない科目の授業を 2 日連続で行ってはならない.

これらの制約は 12747 個の線形不等式制約で記述できる.

この問題例に対して, 1 回の探索の最大計算時間を 300 秒として 30 回探索を行った. 3章で述べた通り, このような大規模な問題例に対しては, タブーリストには, 値変数 x_{ij} よりも変数 X_i を入れておく方が効果的であることが計算実験により確かめられた. tabu tenure t を 5, 10, 20, 30 にそれぞれ固定した場合と自動調節した場合の計算結果を表 3に示す. この問題例には充足不能な制約が 1 つあり, その結果, 最適値は 1 であることが分かっている. 表 3は, 30 回の探索中最適解を得た回数, 平均暫定値, 最悪暫定値, そして t を自動調節した場合には, t の平均値, 及び平均最大値を示している. ここでも, t の自動調節機能の有用性が確かめられ, 現実の時間割問題に対して, 高い確率で最適解を求めていることが分かる.

表 3: 高等学校の時間割問題に対する計算結果

	t : 固定				t : 自動調節
	$t = 5$	$t = 10$	$t = 20$	$t = 30$	
成功率	19/30	25/30	24/30	10/30	26/30
平均暫定値	3.2	2.1	1.3	2.0	1.3
最悪暫定値	20	25	3	4	4
t の平均値	—	—	—	—	9.3
t の平均最大値	—	—	—	—	30.0

7.2 看護婦スケジューリング問題

以下の看護婦スケジューリング問題は東京の病院における実際のデータに基づくものであり, 25 人の看護婦の 1 ヶ月のスケジュールを作成するものである. 各看護婦には, 日勤, 準夜勤, 深夜勤, その他の業務, 休日のいずれか 1 つが毎日割当てられる. 日本の病院における看護婦スケジューリング問題には, 多くの看護婦がどのシフトも受け持ち可能であること, ローテーションの周期が非常に短いこと, そして, 同じメンバーの組合せの繰り返しを好まないことなどの特徴があり, このため, スケジューリングの単位を小さく扱わなくてはならず, 非常に困難な組合せ問題であると言える [11].

この問題を CSP の枠組みで記述するため, $25 \times 30 = 750$ の変数と領域 $D = \{$日勤, ..., 休日$\}$ ($|D| = 5$) を導入する. よって, 値変数の数は $750 \times 5 = 3750$ である. この問題も時間割問題と同様, 多くの制約を含み, その幾つかは問題固有の制約である. 以下に, その幾つかを述べる.

- 25 人の看護婦は A チームと B チーム, それぞれ 13 人と 12 人に分けられ, その中で, A チームと B チームそれぞれ 6 人と 5 人がリーダークラスである.
- 各シフトには, それぞれ必要人数の看護婦, 及び必要人数のリーダーが勤務につかなくてはならない. また, A チームと B チームの人数がほぼ均等にならなくてはならない.
- 各看護婦に対し予め定められているスケジュール (休日など) に違反してはならない.
- 各看護婦に割当てられる各シフトの回数には, 各月ごとにそれぞれ上下限がある.
- 各看護婦は, 少なくとも週に 1 回は, 日勤及び休日をそれぞれ割当てられねばならない.
- 以下のような苛酷な勤務パターンは禁止される. (i) 深夜勤 3 連続, (ii) 準夜勤 4 連続, (iii) 日勤 5 連続, (iv) 深夜勤の次の日に日勤, 準夜勤, またはその他の業務, (v) 準夜勤の次の日に日勤, またはその他の業務, (vi) 1 日だけの深夜勤, (vii) 前後が休日である 1 日だけの勤務.

これらの制約は 9731 個の線形不等式制約で表現することができる. この問題も, 前章で述べた大学の時間割問題のように, 全ての制約を満たす解は存在しないと思われる. そこで, 日勤における必要人数の看護婦, 及びリーダーの確保を考慮制約, それ以外の制約を絶対制約とし, 各ペナルティ関数 $p_l(\boldsymbol{x})$ に対する重み w_l を, 望ましいスケジュールが得られるよう予備的実験により定めた. その結果, 全ての絶対制約を満たす解を幾つか求めることに成功した. そのスケジュールの一例を図 1 に示す. このスケジュールでは, 制約違反として, 日勤におけるリーダーの人数不足が 7 日, 日勤における看護婦全体の人数不足が 1 日ある.

8 まとめと今後の課題

本研究では, 組合せ問題に対する汎用アルゴリズムを目指して, タブー探索アプローチによる CSP アルゴリズムを構築し, 計算実験を行った. その際, プログラムパラメータの自動調節, 目的関数の導入, 近傍の拡張等, 計算性能を高める工夫を組み込み, 現実問題を含め, 幾つかの問題に対してその有用性を確かめた. しかし, 以下のような問題点が残っており, これらを解決していくことが今後の課題である.

1. ある問題を CSP に定式化するとき, その方法は一意ではなく, 定式化の違いは計算結果に大きな影響を及ぼす. よって, CSP への定式化に際しての指針を与えることは重要である. 一般的には, CSP における shift 近傍 (または swap 近傍) が個々の問題においても自然な近傍となるような定式化が望ましい. しかし, 次項による対応も考えられる.

	看護婦	日付																												
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
A	1*	/	−	−	/	≡	≡	/	−	−	/	−	=	/	/	−	=	=	≡	≡	/	/	−	−	=	=	=	/	−	≡	≡
	2*	≡	/	/	−	−	−	=	/	≡	≡	/	+	−	−	/	/	/	−	=	=	≡	≡	/	−	−	=	=	=	/	
	3*	/	/	−	−	=	=	/	≡	≡	/	−	−	/	−	=	≡	≡	/	−	−	/	/	=	≡	≡	/	−	−	−	−
	4*	−	−	=	=	/	≡	≡	/	−	−	/	/	=	=	/	−	−	=	/	+	−	−	−	−	/	/	≡	≡	/	−
	5*	−	≡	≡	/	−	−	=	=	/	−	=	≡	≡	/	/	−	−	−	−	≡	≡	/	/	/	/	−	−	/	−	=
	6*	=	=	/	≡	≡	/	−	−	=	=	/	−	−	≡	≡	/	/	−	=	/	−	=	/	−	−	≡	≡	/	/	−
	7	/	≡	≡	/	−	−	=	/	=	=	=	/	−	−	/	−	−	/	−	−	−	/	−	=	≡	≡	/	−	−	/
	8	=	=	/	/	−	−	/	−	−	+	−	=	/	−	−	−	=	≡	≡	/	−	−	−	−	/	/	−	=	/	/
	9	/	−	−	−	−	=	/	−	−	−	/	/	/	≡	≡	/	−	−	/	−	=	=	/	−	−	−	=	/	−	−
	10	+	/	−	−	/	/	−	≡	≡	/	−	−	−	/	/	+	−	=	=	=	/	−	−	−	=	=	/	≡	≡	/
	11	−	−	=	=	=	/	−	−	/	/	−	−	/	−	−	+	/	−	−	/	/	−	−	/	−	−	=	≡	≡	/
	12	≡	/	/	−	−	−	/	/	/	−	−	−	=	=	/	+	−	/	−	−	−	/	−	−	−	−	/	−	−	≡
	13	−	−	≡	≡	/	−	+	−	/	/	/	−	−	−	=	=	/	−	≡	≡	/	−	=	/	/	/	−	−	=	=
B	14*	=	/	−	=	≡	≡	/	/	−	+	/	/	≡	≡	/	−	−	−	=	=	=	/	−	=	≡	≡	/	−	−	/
	15*	−	=	=	/	−	−	=	/	−	=	≡	≡	/	−	−	−	/	/	/	−	−	=	≡	≡	/	−	−	/	/	−
	16*	≡	≡	/	−	=	/	/	−	≡	≡	/	−	+	/	−	=	≡	≡	/	/	+	−	−	/	=	=	=	/	−	−
	17*	/	/	−	−	−	=	≡	≡	/	−	=	/	+	−	≡	≡	/	−	−	/	/	−	=	/	−	=	≡	≡	/	−
	18*	/	−	≡	≡	/	/	−	−	−	/	/	−	−	/	=	=	=	/	−	+	≡	≡	/	−	−	/	−	=	=	=
	19	−	−	−	−	/	/	+	−	−	−	−	=	/	/	−	+	−	=	/	−	≡	≡	/	/	/	/	−	=	=	=
	20	/	/	−	−	=	/	−	=	/	−	−	≡	≡	/	−	−	=	/	/	−	=	/	−	−	−	−	/	−	−	+
	21	/	−	−	/	/	−	=	/	=	=	=	/	−	−	−	≡	≡	/	−	−	−	/	−	−	−	−	=	/	/	+
	22	+	=	=	/	−	−	−	−	/	≡	≡	/	−	−	/	/	−	−	/	/	−	−	=	=	=	/	−	−	/	+
	23	−	−	−	=	/	−	≡	≡	/	/	−	−	/	=	=	/	/	−	=	=	/	−	−	/	−	−	−	/	−	+
	24	=	/	/	−	−	=	/	/	−	−	−	−	=	/	−	−	/	−	−	/	−	=	≡	≡	/	/	−	−	−	−
	25	−	−	/	/	/	−	−	=	=	/	−	=	=	=	/	/	−	=	≡	≡	/	/	−	−	−	−	/	−	≡	≡
total	−	8	10	10	10	10	11	8	10	9	8	11	10	7	10	10	8	10	10	10	8	9	10	13	10	11	10	10	10	10	8
	=	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	4	4	4
	≡	3	3	4	3	3	3	3	4	3	3	3	3	3	3	3	3	3	3	4	3	3	3	3	3	3	3	3	4	4	3
	+	2	0	0	0	0	0	2	0	0	2	0	0	3	0	0	4	0	0	0	2	1	0	0	0	0	0	0	0	0	4
	/	8	8	7	8	8	7	8	7	9	8	7	8	8	8	8	6	8	8	7	8	8	8	5	8	7	8	7	7	7	6

−: 日勤
=: 準夜勤
≡: 深夜夜
+: その他の業務
/: 休日

*: リーダー

図 1: 計算実験により得られたスケジュールの例.

2. 問題によっては, 問題固有の自然な近傍が定義されることもあり, ユーザが近傍を定義すれば, その近傍を用いた探索が行えるよう, アルゴリズムに柔軟性を持たせることも重要と思われる. 但し, これらの近傍についても, その全ての解を常に調べるのではなく, 何らかの発見的手法により解の候補をふるいにかけ, 計算時間の削減を図る必要があると思われる.

3. 問題のタイプによっては, 係数の大きさ等に影響されて探索が偏ってしまうことがある. 例えば, 線形等式制約において, 係数の分散が大きい場合, 小量の改良を実現するような, 係数の絶対値が小さい変数の値の変更が行われ易く, 多様化を実現するのが困難になる. 本論文のような, tabu tenure t を増加させる方法は, このような問題例に対しては必ずしも有効ではない. このような問題例に対しては, 探索空間の大域的な情報が必要であり, 線形計画法等の, 数理計画的手法が効果的であると思われる.

4. 前処理, あるい探索中に, 問題の規模を縮小したり, 幾つかの変数を固定したりする目的に, CSP の解法として研究されている制約伝播法や backtracking 法を部分的に利用することも考えられる.

5. 6章の大学の時間割問題や7章の看護婦スケジューリング問題のように, しばしば, 制約を絶対制約と考慮制約に分ける必要が生じる. 現在のところ, ペナルティ関数 $p_l(\boldsymbol{x})$ に対する重み w_l を調整することで対処しているが, w_lの調整のために予備的実験が必要であるという欠点がある. これを解決するため, 例えば, 絶対制約を重視した近傍と考慮制約を重視した近傍を 2 つ用意し, 戦略的振動 [9, 10] を行うなどの方法が考えられる.

謝 辞

6章の大学の時間割問題のデータを提供して下さった京都大学の岩間 一雄氏, 7章の高等学校の時間割問題のデータを提供して下さったNEC C&C 研究所の 渡辺 正信氏, 及び 金子 一哉氏, 7章の看護婦スケジューリング問題のデータを提供して下さった成蹊大学の池上 敦子氏に深く感謝致します. なお, 本研究は一部文部省科学研究費によっている.

参考文献

[1] Battiti, R. and Tecchiolli, G., "The reactive tabu search", *ORSA Journal on Computing* 6 (1994) 126-140.

[2] Bowen, J. and Dozier, G., "Constraint satisfaction using a hybrid evolutionary hill-climbing algorithm that performs opportunistic arc and path revision", *Proceedings of the thirteenth National Conference on Artificial Intelligence (AAAI) and the eighth Innovate Applications of Artificial Intelligence (IAAI)* (1996) 326-331.

[3] Davenport, A., Tsang, E., Wang, C.J. and Zhu, K., "GENET: A connectionist architecture for solving constraint satisfaction problems by iterative improvement", *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI)* (1994) 325-330.

[4] Decher, R. and Pearl, J., "Network-based heuristics for constraint-satisfaction problems", *Artificial Intelligence* 34 (1988) 1-38.

[5] Feo, T.A., Venkatraman, K. and Bard, J.F., " A GRASP for a difficult single machine scheduling problem", *Computers and Operations Research* 18 (1991) 635-643.

[6] Fleurent, C. and Ferland, J.A., "Genetic and hybrid algorithms for graph coloring", in: Laporte, G., Osman, I.H. and Hammer, P.L. (eds.), *Metaheuristics in Combinatorial Optimization, Annals of Operations Research* (1996).

[7] Freuder, E.C., "A sufficient condition for backtrack-free search", *Journal of the Association for computing Machinery* 29 (1982) 24-32.

[8] Freuder, E.C., "Complexity of K-tree structured constraint satisfaction problems", *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI)* (1990) 4-9.

[9] Glover, F., "Tabu search - Part I", *ORSA Journal on Computing* 1 (1989) 190-206.

[10] Glover, F., "Tabu search fundamentals and uses", Technical Report, University of Colorado (1995).

[11] 池上 敦子, 丹羽 明, 大倉 元宏, "我が国におけるナース・スケジューリング問題", オペレーションズ・リサーチ 41 (1996) 436-442.

[12] 狩野 均, "制約充足問題の近似解法", 人工知能学会誌 12 (1997) 359-365.

[13] Laguna, M., Feo, T.A. and Elrod, H.C., "A greedy randomized adaptive search procedure for the two-partition problem", *Operations Research* 42 (1994) 677-687.

[14] Minton, S., Johnston, M.D., Philips, A.B., and Laird, P., "Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems", *Artificial Intelligence* 58 (1992) 166-205.

[15] Miyazaki, S., Iwama, K. and Kambayashi, Y., "Database queries as combinatorial optimization problems", *Proceedings of International Symposium on Cooperative Database Systems for Advanced Applications* (1996) 448-454.

[16] 西原 清一, "制約充足問題の基礎と展望", 人工知能学会誌 12 (1997) 351-358.

[17] Nonobe, K. and Ibaraki, T., "A tabu search approach to the CSP (Constraint Satisfaction Problem) as a general problem solver," *European Journal of Operational Research, Special Issue on Tabu Search*, (to appear).

[18] Tsang, E., *Foundations of constraint satisfaction*, Academic Press, London, 1993.

[19] Yoshikawa, M., Kaneko, K., Yamanouchi, T. and Watanabe, M., "A constraint-based high school scheduling system", *IEEE Expert* 11 (1996) 63-72.