1F1-03

動的重み付き制約充足問題に基づく ナーススケジューリングにおける 解の安定性のための制約付加手法の提案

On the Constraint Addition Method for Solution Stability in the Nurse Scheduling based on the Dynamic Valued Constraint Satisfaction Problem

服部宏充[*1] 伊藤孝行[*2] 新谷虎松[*2]
Hiromitsu Hattori Takayuki Ito Toramatsu Shintani

[*1]日本学術振興会特別研究員 JSPS Research Fellow
[*2]名古屋工業大学大学院 工学研究科 Graduate School of Engineering, Nagoya Institute of Technology

Scheduling has been an important research field in Artificial Intelligence. Because typical scheduling problems could be modeled as Constraint Satisfaction Problems (CSP), several constraint satisfaction techniques have been proposed. In order to handle the different levels of importance of the constraints, solving the problems via Valued Constraint Satisfaction Problems (VCSP) is an promising approach. However, there exists the case where an unexpected events which might require a sudden change in the obtained schedule, i.e., the case with dynamic changes in scheduling problems. In this paper, we describe such dynamic scheduling problem as Dynamic Valued Constraint Satisfaction Problems (DyVCSP) in which constraints would changes dynamically. Generally, it is undesirable to determine vastly modified schedule even if re-scheduling is needed. A new schedule should be close to the current one as much as possible. In order to obtain stable solutions, we propose the methodology to maintain pieces of the current schedule via the use of temporal soft constraints, which is explicitly penalizing the changes from the current schedule. In this paper, we focus on the nurse scheduling for applying our method.

1. はじめに

制約充足問題（Constraint Satisfaction Problems:CSP）[7] は，変数間に存在する制約の充足を条件として変数に割り当てる値を決定する問題であり，分散問題解決，およびマルチエージェントに関連する重要な研究分野である．CSP の応用分野としてスケジューリング問題がある．実世界におけるスケジューリング問題は，多くの場合で過制約となり，全ての制約を充足することが困難となる．過制約の問題を扱うアプローチとして，重み付き制約充足問題（Valued Constraint Satisfaction Problem: VCSP）に基づいて求解を試みる方法が盛んに研究されている [1, 4]．VCSP では，各制約に対して，その制約が違反された場合のコストを表す重みが定義され，違反する制約の重みの和を最小化する解を求めることが目標となる．本論文では，過制約なスケジューリング問題を VCSP として定式化し，解の決定を行う．

処理が困難なケースとして，新たなイベントが突発的に追加されることで，決定済みのスケジュールに対して変更が必要とされる場合がある．この場合，追加されたイベントを含めた再スケジューリングが必要となる．つまり，時間の経過とともに，スケジューリング問題自体が動的に変化する．本論文では，動的に変化するスケジューリング問題を動的重み付き制約充足問題（Dynamic Valued Constraint Satisfaction Problem）として定式化する．動的 VCSP は，変数やその値，もしくは制約の増減により，時間と共に断続的に変化する VCSP である．しかし，動的 VCSP では，問題のわずかな変化によって，以前に決定されたスケジュールが大幅に変更される可能性がある．スケジュールの大幅な変更は，一般には望ましくなく，新たに決定されるスケジュールは，以前のスケジュールに近似している方が良い．動的 CSP において，変数に対して割り当てる値の変更を抑えるという条件は，解の安定性と呼ばれる [6, 7]．本論文では，動的 VCSP において安定した解（スケジュール）を得るための，制約付加手法を提案する．本手法では，暫定的な制約を利用することで，以前に決定されたスケジュールの維持を試みる．ここで用いる暫定的制約は，各変数に対して，割り当てられた値をできる限り維持するための制約である．これらの制約は，他のスケジューリングのための制約とは区別され，再スケジューリングのプロセスが終了した後に削除される．

本論文では，対象とする具体的な問題として，病院におけるナーススケジューリングに注目する．ナーススケジューリングでは，重要度の異なる様々な制約を考慮に入れる必要がある．例えば，各看護師からの希望だけでなく，法的な規制や，人員配置数に関する規則などを考慮しなければならない [2]．ナーススケジューリングでは全ての制約を同時に充足することは困難であるため，制約の重要度を明示的に表現し，より重要な制約の集合を満足するための計算を行う必要がある．ここでは，看護師からの休暇申請などの突発的な要望が発生した場合に，再スケジューリングを行う必要がある．看護師の混乱を避けるため，新たに決定されるスケジュールは，直前のスケジュールからの大幅な変更の無い，安定したものであることが望まれる．以上により，本論文では，ナーススケジューリング問題を動的 VCSP として定式化し，提案する制約付加手法を用いた安定したスケジュールの決定を実現する．

本論文の構成は以下の通りである．まず 2. では，本論文で導入する動的 VCSP について述べる．3. では，本論文で対象とするナーススケジューリング問題について述べ，定式化を行う．4. では，解の安定性のための制約付加手法について述べ，具体的なスケジューリングのプロセスを示す．5. では，提案手法について議論し，6. で結論と今後の課題を述べる．

連絡先: 服部 宏充，名古屋工業大学大学院 工学研究科，〒 466-8555 愛知県名古屋市昭和区御器所町，TEL: (052)735-7330，FAX: (052)735-5584，E-mail:hatto@ics.nitech.ac.jp

2. 動的重み付き制約充足問題

本論文では，実世界におけるナーススケジューリング問題を扱う際に，過制約，もしくは解が存在しないケースを効果的に処理するために，ある時点での制約集合に基づく個々のスケジューリング問題を VCSP として定式化する．VCSP では，各制約に対して重みを定義するため，その重要度を明示的に表現／活用できる．さらに，動的 CSP の考え方の導入に基づいて VCSP を拡張し，動的 VCSP として問題を定式化することで，看護師からの要望による問題の動的変化を処理可能とする．動的 VCSP は，通常の VCSP の列として表現される．タイムステップ i における VCSP を $\mathcal{VP}_i$ とすると，動的 VCSP（$\mathcal{DP}$）は以下のように表現できる．

$$\mathcal{DP} = \{\mathcal{VP}_0, \mathcal{VP}_1, ..., \mathcal{VP}_i, ...\}$$

ここで，$\mathcal{VP}_{i+1}$ は，直前の問題 $\mathcal{VP}_i$ に関して，変数やその値，もしくは制約の増減が発生することで生成された問題となる．

個々の VCSP は，$\mathcal{VP}_i = (X_i, D_i, C_i, S_i, \varphi_i)$ と表現される．(X_i, D_i, C_i) は従来の CSP であり，それぞれ変数，各変数の値域，および制約の集合を表す．$S = (E, \otimes, \succ)$ は，評価構造である．$\varphi : C \rightarrow E$ は評価関数であり，各制約に対して評価値を与える．E は可能な評価値の集合であり，$\succ$ は E の要素に全順序関係を定義する．また，最大／最小の評価値を，それぞれ $\top \in E$，および $\bot \in E$ とする．$\otimes$ は評価値の結合演算を定義する．ここで，$\mathcal{A}$ を全変数への値の割り当てとすると，制約 c に関する割り当て $\mathcal{A}$ の評価値は以下のように定義される．

$$\varphi(\mathcal{A}, c) = \begin{cases} \bot & \text{if } c \text{ is satisfied by } \mathcal{A} \\ \varphi(c) & \text{otherwise} \end{cases}$$

そして，全ての制約を考慮した場合の割り当て $\mathcal{A}$ の評価値は，

$$\varphi(\mathcal{A}) = \otimes_{c \in C} \varphi(\mathcal{A}, c).$$

となる．

3. 動的重み付き制約充足問題に基づくナーススケジューリング

3.1 ナーススケジューリング問題

ナーススケジューリング問題とは，複数の看護師に，一定期間のワーキングシフトを割り当てる問題である．制約としては，法的規則，組織のルール，最小人員配置数，看護師の休暇，および看護師の希望なとが考えられる．看護師の勤務形態として，3 交代制の勤務形態がある．3 交代制のもとでは，1 日が 3 つの勤務シフト（日勤／準夜勤／夜勤）に分けられ，各看護師には，休暇の場合と合わせて 4 通りのスケジュールのいずれかが割り当てられることになる [2].

本論文におけるナーススケジューリング問題は，看護師，スケジューリングの対象期間，および 3 交代制に基づくワーキングシフトに基づき，要求された制約をより多く満たすスケジュールを生成することが目的である．ここでは，各看護師あたりの勤務日数，各シフトあたりの要求人員数，看護師の希望等を制約として考慮する．また，各制約には，重要度を表す重みが付加される．ここで，法的規則には最大の重みを付加することとし，求解の過程で必ず充足されるようにする．スケジューリングの結果，スケジューリングの対象期間の日付における各看護師のワーキングシフトを示すテーブルが生成されることになる．

3.2 動的重み付き制約充足問題に基づく定式化

本節では，3.1 で述べたナーススケジューリング問題を動的 VCSP として定式化する．本論文が対象としている，動的に変化するナーススケジューリング問題は，VCSP のシーケンスとして定義でき，かつ個々の VCSP は，各タイムステップにおける，異なる変数とその値，および制約から成る．タイムステップ i のスケジューリング問題を表す VCSP $\mathcal{VP}_i$ を，$\mathcal{VP}_i = (X_i, D_i, C_i, S, \varphi)$ と定義する．変数の集合 $X_i = \{x_{(1,1)}, x_{(1,2)}, ..., x_{(s,t)}\}$ の各要素は，各日付における各看護師のスケジュールを表すとする．例えば，要素 $x_{(s,t)} \in X_i$ は，日付 t における，看護師 s のスケジュールを表す．D_i は変数の値域集合を表すが，本ナーススケジューリング問題においては全変数の値域は共通とし，例えば，変数 $x_{(s,t)}$ の値域は，$d_{(s,t)} = 0, 1, 2, 3 \in D_i$ とする．ここで取り得る値はそれぞれ，0 = 休暇，1 = 日勤，2 = 準夜勤，および 3 = 夜勤を表す．また，評価構造 $S = (E, \otimes, \succ)$ に関して，E は整数の集合であり，$\bot = 0$，かつ $\top = 9$ である．また，$\succ$ は通常の全順序関係として，$\otimes = \Sigma$ とする．従って，全変数への完全な値の割り当てを $\mathcal{A}$ とすると，$\varphi(\mathcal{A})$ は違反制約の重みの和となる．

制約集合 C_i に含まれる制約の形式を以下のように定義する．

$$lim(min, max, List, w) \quad (1)$$

ここで，min および max は，タイムステップ i における全変数に割り当てられた値の集合 $\mathcal{A}_i$ と $List$ を比較して，一致する要素数の下限 min と上限 max を表す．$List$ には，制約が要求する任意の変数の値の集合である．w は制約の重みであり，0〜9 の整数値を取る．制約は，$\mathcal{A}_i$ と $List$ の間で一致する要素数 n が，$min \leq n \leq max$ である場合にのみ充足される．例えば，ある看護師 s が，スケジューリングの対象期間中に，休暇を 1 回以上，3 回以下得られることを希望する場合の制約は以下のように記述できる．

$$lim(1, 3, \{x_{(s,1)} = 0, x_{(s,2)} = 0, ..., x_{(s,t)} = 0, ...\}, 5) \quad (2)$$

つまり，この制約は，0 の値を取る変数の数が 1 以上 3 以下である場合に充足されることを意味している．また，この制約が違反された場合は 5 のコストが加算される．

4. 制約付加に基づく再スケジューリング

4.1 制約付加に基づく解の安定性の実現

前節までの定式化により，VCSP の列 $\{\mathcal{VP}_0, \mathcal{VP}_1, ..., \mathcal{VP}_i, ...\}$ から成る動的 VCSP として，ナーススケジューリング問題を表現できる．$\mathcal{VP}_i$ は，通常の VCSP として解くことができる．通常，スケジューリング問題 $\mathcal{VP}_i$ と，$\mathcal{VP}_i$ に変更が加えられて生成された問題 $\mathcal{VP}_{i+1}$ は，解の計算に関して無関係であり，互いに影響を受けない．つまり，各問題は，個々に独立して求解が可能である．従って，$\mathcal{VP}_i$ と $\mathcal{VP}_{i+1}$ のように，前後の関係にある問題でも，得られるスケジュールに影響が生じることはなく，逆に互いに大きく異なる場合も起こり得る．動的 VCSP における解の安定は，VCSP の列 $\{\mathcal{VP}_0, \mathcal{VP}_1, ..., \mathcal{VP}_i, ...\}$ の個々の問題を逐次的に解いていく過程を通して解が大幅に変化しないことで実現される．従って，以前に得られた解を何らかの方法で利用することで，VCSP 間に相互依存関係を発生させ，安定した解を得る方法が必要と考えられる．

以上の考察から，本論文では，各変数に対して，以前の問題の解において割り当てられた値と同じ値を持つ，という暫定

的制約（provisional constraint）を導入する手法を提案する．つまり，以前に得られた解自体を得るための重み付き制約を付加する．たとえば，以前の問題において変数 x_{ij} に割り当てられた値を v_{ij} とすると，以下の暫定的制約が追加される：

$$lim(1, 1, \{x_{ij} = v_{ij}\}, w) \tag{3}$$

ここで，w は，事前に定められた暫定的制約に対する重みである．

本手法では，直前に得られた解の近傍の解だけでなく，より以前に得られた解の近傍の解を得ることを可能する．従って，暫定的制約の集合を C_{prov} とすると，$|\ C_{prov}\ |$ は，問題が変化し，再スケジューリングが行われるにつれて単調に増加する．本手法により，動的 VCSP において，複数の VCSP を逐次的に解いていく過程を通して解の安定性を実現できる．

ただし，本手法では，看護師の要望に基づいて新たに追加される制約は，暫定的制約よりも高い重みを持つ必要がある．新たに追加される制約の重みが暫定的制約よりも小さい場合，再スケジューリングによって，明らかに直前の問題と同一の解が得られ，再スケジューリングは無意味となる．

4.2 再スケジューリングのプロセス

本節では，4.1 で示した制約付加手法に基づくスケジューリングプロセスを示す．ここでは，看護師からの要望に基づく制約の集合 C_{new} の追加により，スケジューリング問題 $\mathcal{VP}_i$ が，新たな問題 $\mathcal{VP}_{i+1}$ へと変化している．以下に，4 つのステップから成る再スケジューリングのプロセスを示す．

Step 1: 看護師の要望から生成された制約の集合 C_{new} を $\mathcal{VP}_i$ に追加し，新たなスケジューリング問題 $\mathcal{VP}_{i+1}$ を生成する．ここでは，再スケジューリングを行う日付以前のスケジュールを変更不能とするため，該当する変数に関して，同一の値を取るための重み 10 の制約が生成／追加される．

Step 2: 全ての変数に関して，問題 $\mathcal{VP}_i$ を解いて得られたスケジュールと同一の値を取るための暫定的制約 C^i_{prov} を生成し，それまでのスケジューリングの過程で得られた暫定的制約の集合 C_{prov} に追加する．従って，

$$C_{prov} = \bigcup_{j=0}^{i} C^j_{prov} \tag{4}$$
$$(\text{ただし}, \forall j\ c \in C^j_{prov}, c \notin C_{prov})$$

となる．

Step 3: 暫定的制約の集合 C_{prov} を $\mathcal{VP}_{i+1}$ に追加し，仮の問題 $\mathcal{VP}'_{i+1}$ を生成する．ここで，暫定的制約 C^i_{prov} に含まれる制約の重みの合計値 $W_{C^i_{prov}}$ は，Step 1 における C_{new} に含まれる制約の重みの合計値 $W_{C_{new}}$ を越えない場合にのみ次のステップに進む．もしも，$W_{C_{new}} \leq W_{C^i_{prov}}$ である場合，$\mathcal{VP}_i$ を解いて得られたスケジュールが維持されるため，再スケジューリングは無意味である．

Step 4: Step 3 で生成された問題 $\mathcal{VP}'_{i+1}$ に対して，基本的な山登り探索を適用し，スケジュールの決定を行う．ここで，Step 2 における暫定的制約の追加によって解の安定性は保証されるため，新たな VCSP である $\mathcal{VP}'_{i+1}$ を最初から解くことで，可能な限り変更の割合を押さえたスケジュールを決定できる．

$\mathcal{VP}_i = (X_i, D_i, C_i, S, \varphi)$: VCSP at time step i
$\mathcal{A}_i$: an assignment for $\mathcal{VP}_i$
C_{new}: a set of constraints which is added to $\mathcal{VP}_i$
C_{prov}: a set of provisional constraints for solution stability
w: the weight of provisional constraint which is predefined

```
1:  re_scheduling (VP_i, A_i, C_new, C_prov)
2:    C_{i+1} ← C_i ∪ C_new
3:    ID ← ID of nurse associated with c ∈ C_new
4:    DT ← date associated with c ∈ C_new
5:    for each x_(s,t) ∈ X_i
6:      if s ∉ ID ∨ t ∉ DT then
7:        C_prov ← C_prov ∪ lim(1, 1, {x_(s,t) = v_(s,t)}, w)
8:    end for
9:    C'_{i+1} ← C_{i+1} ∪ C_prov
10:   A_{i+1} ←solve_VCSP(X_i, D_i, C'_{i+1}, S, φ)
11:   for each c ∈ C_prov
12:     if (c is not satisfied in A_{i+1}) then
13:       C_prov ← C_prov − {c}
14:   end for
15:   VP_{i+1} ← (X_i, D_i, C_{i+1}, S, φ)
16:   return VP_{i+1}, A_{i+1} and C_prov
```

図 1: 再スケジューリングのアルゴリズム

Step 5: 解の安定性のために追加された C_{prov} を取り除き，問題を $\mathcal{VP}'_{i+1}$ から $\mathcal{VP}_{i+1}$ へ戻す．また，$\mathcal{VP}'_{i+1}$ を解いた際に充足された暫定的制約を C_{prov} から取り除く．これは，新たな問題 $\mathcal{VP}_{i+2}$ を解く必要が生じた場合に，同一の暫定的制約の重複を防ぐためである．

タイムステップ i における，再スケジューリングアルゴリズムの詳細を図 1 に示す．第 1 行では，再スケジューリングを行う関数 *re_schedule* への入力として，タイムステップ i におけるナーススケジューリング問題 $\mathcal{VP}_i$，VP_i における全変数への値の割り当て A_i，新たに追加された制約の集合 C_{new}，およびこれまでに生成された暫定的制約の集合 C_{prov}，の 4 つが与えられている．

第 5 行から第 8 行にかけて，問題 $\mathcal{VP}_i$ から得られたスケジュールにおいて，各変数へ割り当てられた値 $v_{(s,t)}$ を維持するための暫定的制約を新たに生成し，C_{prov} へ格納している．ここで，C_{new} 中の制約によって変更が試みられている変数に関しては，暫定的制約は生成しない．図 1 に示したアルゴリズムでは，暫定的制約の重みは全て同一で事前に定義されているが，必要に応じて異なる重みを与えることも可能である．

第 10 行では，基本的な山登り探索に基づいて VCSP を解く関数 $solve\_VCSP$ により，タイムステップ i における各変数への値の割り当てを決定する．ここで，関数 $solve\_VCSP$ の引数 $(X_i, D_i, C'_{i+1}, S, \varphi)$ は，$\mathcal{VP}'_{i+1}$ を表す．

第 11 行から第 14 行では，得られたスケジュール $\mathcal{A}_{i+1}$ において充足されなかった暫定的制約のみを取りだし，最終的に C_{prov} に格納している．

図 2 では，本論文で，VCSP を解く過程で用いる山登り探索のアルゴリズムを示している．本論文では，基本的な山登り探索のアルゴリズムに，文献 [6] と同様のランダム性を導入している．ここでは，まず，変数の集合 $X = x_0, x_1, ...$ と，各変数に割り当てる値の集合 $V = v_1, v_2, ...$ から成る解 $s(X, V)$

$\mathcal{VP}_i = (X_i, D_i, C_i, S, \varphi)$: VCSP at time step i

```
1:  hill_climbing (VP_i)
2:    s(X,V) ← set_initial(VP_i)
3:    V_best ← V
4:    repeat
5:      if eval(s(X,V)) = 0 then
6:        return s(X,V)
7:      for i = 0 to | X |
8:        with probability p:
9:          v'_i ← arg max_{v∈d_{x_i}} eval(s(X, V ∪ {v} − {v_i}))
10:       with probability 1 − p:
11:         v'_i ← a value chosen at random in d_{x_i}
12:       V' ← V ∪ v'_i − {v_i}
13:       if eval(s(X,V')) ≤ eval(s(X,V_best)) then
14:         V_best ← V'
15:     end for
16:     V ← V_best
17:   until cutoff steps
18: return s(X,V)
```

図 2: 山登り探索のアルゴリズム

に対して，関数 *set_initial* で初期の暫定解を設定する（第 2 行）．そして，違反制約が皆無の解を発見するか，事前に設定したループ数（cutoff steps）に達するまで，違反制約が少なくなる方へ解を改善する．ここで，変数に値を割り当てる際に，確率 p で最も制約違反数が小さくなる最良の値を割り当て（第 7 行から第 8 行），確率 $1-p$ でランダムに選択した値が割り当てられる（第 9 行から第 10 行）．ただし，いずれの場合においても，$v'_i \neq v_i$ である．また，d_{x_i} は，変数 x_i の値域を表す．

5. 議論

本節では，本論文で提案した，解の安定性を実現するための暫定的制約の付加手法について，その有用性と妥当性を関連研究を交えて議論する．

動的 CSP における解の安定性を実現するためのアプローチとして 2 つの方法が挙げられる．ひとつは，初期値利用法と呼ばれるもので [5]，直前の問題の解を，次の問題における初期値として利用し，段階的にその値を改善して行く方法である．もうひとつは，制約記録法と呼ばれるもので [3]，以前の問題の解を求める過程で新たに導かれた制約と，その制約が成立するための前提条件を記録しておく方法である．これらのアプローチは，解の安定性を実現するために，CSP の解決アルゴリズムを工夫するものである．一方，本論文で提案した手法は，これらのアプローチとは異なり，解くべき問題そのものを変形し，解の安定性を実現している．4.2 に示したプロセスでは，新たなスケジューリング問題 $\mathcal{VP}_{i+1}$ に対して，以前得られた解の近傍で解を決定するために，変数の値の変更を抑えるための暫定的制約を加えた仮の問題 VP'_{i+1} を生成している．解の安定性を得るために，解くべき問題を適切に変形する本論文のアプローチは実現が容易であり，かつ個々の VCSP を解くアルゴリズムの選択を制約しないため，応用がしやすい．

暫定的制約を追加して求解を試みる本提案手法では，考慮すべき制約の数を増加させている．さらに，暫定的制約の数は，再スケジューリングを重ねるごとに単調に増加し続けるため，一般的には，本論文で述べたような山登り探索による最適な解の決定は徐々に困難になると予想される．しかし，文献 [8] では，制約数が一定の数を越えることで，逆に良質の解に到達しやすくなる現象が生じる事が報告されており，制約を増加させる提案手法は，スケジューリングの回数を重ねることで，より良く機能する可能性があると考えられる．

6. おわりに

本論文では，過制約かつ動的な変化を生じ得るスケジューリング問題を動的重み付き制約充足問題として定義し，ここでの解の安定性を実現するための制約の付加手法を提案した．また，具体的な応用分野として，本論文では，病院におけるナーススケジューリング問題に注目し，動的重み付き制約充足問題としての定式化を示し，提案手法の応用が可能であることを示した．提案手法は，既存の他の手法とは異なり，スケジューリング問題自体を変形し，通常のアルゴリズムに基づいて解くだけで安定した解が得られる．また，解の安定性を実現するために，余分に制約を考慮する必要があるが，文献 [8] の報告から，本手法は十分に妥当であると考えられる．

参考文献

[1] Abdennadher, S., Schlenker, H.: Nurse scheduling using constraint logic programming, Proc. of the 11th Annual Conference on Innovative Applications of Artificial Intelligence (IAAI-99), pp. 838–843, 1999.

[2] 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，Vol. 41，No. 8，pp. 436–442，1996.

[3] Schiex, T., Verfaillie, G.: Nogood recording for static and dynamic constraint satisfaction problem, International Journal of Artificial Intelligence Tools, pp. 187–207, 1994.

[4] Tsuruta, T., Shintani, T.: Scheduling meetings using distributed valued constraint satisfaction algorithm, Proc. of the 14th European Conference on Artificial Intelligence (ECAI-2000), pp. 383–387, 2000.

[5] Verfaillie, G., Schiex, T.: Solution reuse in dynamic constraint satisfaction problems, Proc. of the 20th National Conference on Artificial Intelligence(AAAI-94), pp. 307–312, 1994.

[6] Wallace, R.J., Freuder, E.C.: Stable solutions for dynamic constraint satisfaction problems, Proc. of the 4th International Conference on Principles and Practice of Constraint Programming, pp. 447–461, 1998.

[7] 横尾真，平山勝敏: CSP の新しい展開：分散／動的／不完全 CSP，人工知能学会誌，Vol. 12，No. 3，pp. 33–41，1997.

[8] Yokoo, M.: Why Adding More Constraints Makes a Problem Easier for Hill-climbing Algorithms: Analyzing Landscapes of CSPs, Proc. of the 3rd International Conference on Principles and Practice of Constraint Programming (CP-97), pp. 356–370, 1997.