アルゴリズム 32−13
（1993. 3. 18）

CNF論理式に対する局所探索法の評価

宮崎 修一　岩間 一雄

九州大学工学部

局所探索法とは，CNF論理式のある（ランダムな）セル（変数への0,1のある割り当て）から始め，そのセルの近傍のセルを眺めて，それを覆っている節が少ない方へと進んでいくことによって解を見つけるアルゴリズムである．非常に最近，我々を含む少なくとも3つのグループによってほぼ同時に発見され話題になった．本稿では，本アルゴリズムの実験的及び解析的性能評価を以下の3種類の入力に対して行なう．(i) ランダムCNF式．(ii) 局所探索法が不得意とするような人工的CNF式．(iii) ハミルトン閉路問題を変換したCNF式．特に，(iii) に対しては変換を工夫することにより，15頂点程度まで実用的時間で解けることが判った．

キーワード：SATアルゴリズム，局所探索，ハミルトン閉路問題

The local search algorithm for CNF equations and its performance

Shuichi Miyazaki and Kazuo Iwama

Department of Computer Science and Communication Engineering
Kyushu University

Fukuoka 812, Japan

The local search algorithm for CNF predicates was found very recently by at least three different research groups including the present authors. The algorithm starts from a randomly selected cell (an assignment of true and false to variables), looks at its neighbors and moves to the one which is covered by a less number of clauses. In this paper, experimental and analytical evaluation of this algorithm's performance is made for following three different kinds of inputs : (i)Random CNF predicates (ii)Artificially made predicates so that the algorithm will not work well (iii)Predicates reduced from the Hamiltonian Circuit Problem(HCP). In terms of (iii), a sophisticated reduction is shown, by which HCP can be solved in reasonable time up to 15 vertices.

Keywords: SAT algorithms, local search, Hamiltonian Circuit Problem

1. はじめに

CNF（和積形）論理式の充足可能性問題（SAT）とは，与えられた論理式を1にする変数の組合せが存在するかどうかを問う問題である．SATは最も基本的な組合せ問題の一つであり，最初に証明されたNP完全問題としても有名である．SATは計算機上での様々な問題を，それに帰着させて解くという実用的な面ばかりか，理論的な面からも極めて重要な問題であり，古くから効率良く解くためのアルゴリズムが研究されている．このアルゴリズムのほとんどはバックトラック法に基づくものである[1][4][8][9][10]．また近年，斬新な手法を用いたものも考えられている[2]．これは論理式がカルノー図上で囲むセルの数を重複を除いて数えあげ，全てのセルを囲めば充足不能，囲まれていないセルがあれば充足可能という判定を下すものである．

最近，我々を含む世界の異なる3つのグループによって局所探索法とよばれる新アルゴリズムがほぼ同時に発見された[5][6][7]．局所探索法とはカルノー図上でオーバーラップ数の少ない方へと進んでいくことによってオーバーラップ数0のセル（すなわち充足解）にたどりつくものである．

以下では，まず，このアルゴリズムを考えるに至った基本的な着眼点を述べるとともに，アルゴリズムを紹介する．次に，ランダムに生成された例題に対して，どのような性能を発揮するかを計算機実験により調べる．この実験は，局所探索法と，$x_0,\cdots,x_{n-1}$ にランダムに0，1，を代入して，それが充足解になっているかどうか調べるランダム探索法に対して行ない，その結果を比較するという形で行なわれた．効率は局所探索法の方が10変数の場合で，5倍～20倍，20変数の場合で，4倍～100倍優れていた．

次に，局所探索法が解きにくいと思われる例題を生成するアルゴリズムを考え，それによって生成された例題は，局所探索法には非常に解きにくい例題であることを数学的な解析によって示す．局所探索法は直観的にオーバーラップの傾斜がなだらかなときに効率がよい．よって本生成法では，解を1つ決め，それ以外のセルを節で覆いつくしてしまう．そして，解の隣を必ず囲むような節を生成することによって解の周りのオーバーラップ数を多くして，局所探索法では解の周りに近づけないようにしている．

最後に，より実際的な例題に対する局所探索法の効率を調べるために行なった計算機実験の結果を述べる．同じNP完全問題であるハミルトン閉路の例題をSATに変換したものを入力として与えた．はじめに，ハミルトン閉路の例題のグラフの頂点数，枝数をそれぞれ N，M として NM 変数でSATに変換した場合に対して実験を行なった．この場合，論理式が非常に偏った形のものとなり，実験結果によれば局所探索法にとって都合の悪い場合になることが分かる．しかし，変換方法を工夫すれば，この偏りをかなり減らすことが可能である．$N(\frac{1}{k}logN+\frac{k-1}{k}logD)$ 変数による変換法（ただし，D はグラフの最大次数，k は定数）を紹介し，この場合15頂点程度までは実用的時間で解けることを示す．このように，ある問題を別の問題に変換して実際に解こうとしたときは，その変換の効率が非常に重要になる．従来NP完全性の証明等ではそのような変換は単に多項式時間であればよいとされていた．今後は実用化をも視野に入れ，より精密な変換を追求する努力が重要になってくると考えられる．

2. 局所探索法

2.1 CNF充足可能性問題

CNF充足可能性問題の例題は，CNF論理式として与えられる．CNF論理式とは，節の積であり，節とはリテラルの和である．リテラルとは変数 $x_i(i=1,2,\cdots,n)$ の肯定 x_i 及び否定 $\overline{x_i}$ である．節は $c_j(j=1,2,\cdots,m)$ で表される．節 c_j はその中に1（真）となるリテラルが1つでもあるとき1になり，すべてのリテラルが0（偽）のとき0となる．CNF論理式 f はそのすべての節が1となるような変数の値の組合せが存在するとき充足可能（YES）であるといい，そういう組合せが存在しないとき充足不能（NO）であるという．

SATを解くためのアルゴリズムとしてこれまでに主に研究されてきたアルゴリズムは，バックトラック法という考えに基づくものが多い．バックトラック法とは，基本的には次のようなアルゴリズムである．

ステップ1： 変数 $x_1,x_2,\cdots,x_n$ をある順番に並べ，$x_{i1},x_{i2},\cdots,x_{in}$ とする．

ステップ2： $j=1$ とする．

ステップ3： x_{ij}=1として，式を簡単化する．式が0または1となればステップ4へ．式が0にも1にもならなかったら，$j=j+1$ としてステップ3へ．

ステップ4： x_{ij}=0として式を簡単化する．式が0または1となればステップ5へ．式が0にも1にもならなかったら $j=j+1$ としてステップ3へ．

ステップ5： x_{ij}=1である変数 x_{ij} を見つけるまで $j=j-1$ とする．そして x_{ij}=0として式を簡単化する．式が0または1となればステップ5へ．式が0にも1にもならなかったら $j=j+1$ としてステップ3へ．

バックトラック法では，値を代入する変数を選ぶ順番や，変数に1，0どちらを先に代入するかによって効率が大きく異なる．したがって，変数や値の選び方の違う様々なアルゴリズムが考え出されている．例えばUnit Clause Backtracking[10]では，中に1つのリテラルしかない節（Unit Clause）を見つけて，そのリテラルが1になるように値を代入し，式を簡単化するものである．また，Pure Literal rule[9]では論理式の中に肯定または否定のリテラルだけしか現れていない変数があれば，そのリテラルが1になるように値を代入していくものである．

また，これとは別にカルノー図をイメージしたアルゴリズムが最近考え出された[2]．これは次のような考えに基づいている．カルノー図上で，全ての変数に値をいれた時にその組合せを表す小片をセルとよび，あるセルを囲んでいる異なった和項の個数を，そのセルのオーバーラップ数と言うことにする．論理式をカルノー

図 1: カルノー図

図上に表した時，オーバーラップ数が 0 のセルが充足解である．例題の変数が n 個であれば，セルの数は 2^n 個ある．また，節中に異なる変数のリテラルが s 個あればその節はカルノー図を 2^{n-s} 個囲む．次に，オーバーラップの重複について考える．クローズ c_i に x_k，節 c_j に $\overline{x_k}$ が含まれるという x_k が 1 つでも存在する場合，または i と j を逆にした場合には c_i と c_j は同じセルを囲まない．そういう x_k が 1 つも存在しない場合には，c_i と c_j は同じセルを囲む．このとき c_i に含まれるリテラルの集合を A_i，c_j に含まれるリテラルの集合を A_j とし，$A_i \cup A_j$ の要素の数を p とすれば c_i と c_j は 2^{n-p} 個のセルを重複して囲んでいることになる．以上のように考えて，論理式が囲むセルの数を重複を除いて数え，それが 2^n であれば論理式は全てのセルを覆っていることになり充足不能である．また，2^n でなければ囲まれていないセルがあるので，それが充足解となり充足可能である．

2.2 局所探索法

局所探索法ではカルノー図上でのオーバーラップの傾斜に注目している．つまり与えられた論理式をカルノー図上で表現した場合に，オーバーラップ数の分布がなだらかな傾斜になっていれば，オーバーラップ数の多いセルの周りにはあまり解はなく，オーバーラップ数の少ないセルの周りに解が存在している可能性が高いと考えられる．従って，注目しているセルから周りを見て（局所探索），1 番オーバーラップ数の少ないセルへ注目の対象をずらす動作を繰り返せば，だんだんオーバーラップ数の少ない領域，即ち解の存在する可能性の高い領域へと進んでいき，うまく解にたどりつくことができるのではないかとの観測に基づいている (図 1).

アルゴリズム 局所探索法

ステップ 1： ランダムに初期値 $X = (x_0, \ldots, x_{n-1})$ を決める．

ステップ 2： X のオーバーラップ数を計算し，$P(X)$ とする．

ステップ 3： $P(X) = 0$ なら，充足可能．$P(X) \neq 0$ なら，ステップ 4 へ．

ステップ 4： 1 つの変数だけが X と異なるもの $X_0, \ldots, Xn-1$ $(X_i = (x_0, \ldots, x_{i-1}, \overline{x_i}, x_{i+1}, \ldots, x_{n-1})\ i = 1, \ldots, n-1)$ を計算し，それぞれのオーバーラップ数 $P(X_i)$ を計算する．

ステップ 5： $P(X_i)$ の最小値を選び（複数個ある場合はランダムに選ぶ），その時の X_i を Y とする．$P(X) > P(Y)$ なら $X := Y$ としてステップ 2 へ．$P(X) \leq P(Y)$ ならステップ 1 へ．

このアルゴリズムの全体的な流れとしては，まず初期値となるセルをランダムに選び，そこから周りを見て自分よりもオーバーラップ数が少なくその中でオーバーラップ数が最小であるセルに移動し，そこからまた周りを見るという操作を繰り返していくものである．もし，周りのオーバーラップ数が全て自分以上だった場合は，新たに初期値を選び直すことにする．

各ステップの動作は，まずステップ 1 でランダムに初期値を選びそれを最初の注目の対象 (X) とし，ステップ 2 で X のオーバーラップ数を計算している．ステップ 3 では，X が充足解であるかどうかの判定を下している．すなわち，X のオーバーラップ数が 0 であれば充足解であるが，0 でなかったら解ではないということである．X が充足解でなかったら，ステップ 4 で X に隣り合うセルのオーバーラップ数を計算する．そしてステップ 5 で，それらが X のオーバーラップ数より大きいかどうかを判断している．

このアルゴリズムはその性質からいって，全てのセルを見尽くすことはできない．したがって答が NO であるという結論を下すことはできない訳である．

3. 多様な例題に対する局所探索法の評価

3.1 ランダム生成例題

局所探索法の効率を調べるために，ランダム生成例題に対する計算機実験を行なった．ランダム生成例題とは，あるパラメーターに従って生成された例題であり，より平均的な例題であると言える．これは，アルゴリズムの総合的な評価をするためにふさわしい例題であると考えられている．

実験に当たっては，上述の局所探索法と，ランダム探索法 ($x_0, \ldots, x_{n-1}$ にランダムに 0，1，を代入して，それが充足解になっているかどうか調べる）の 2 つに対し，それぞれに例題を与えて [3]，解を探し出すまでの探索回数を比較した [7]．ランダム生成され，かつ，解の存在する例題のパラメーターをいろいろと変えて 100 回の平均をとった．パラメーターとしては，変数の数 v，節の数 t，節に現れる変数の割合 p を用いた．結果は表 1 ～表 6 に示す通りである．

n.a. のところは時間がかかり過ぎて測定できなかったことを示す．この結果からわかるように，効率は 10 変数の場合は局所探索法はランダム探索法の 5 倍～ 20 倍，20 変数の場合で，4 倍～ 100 倍となった．局所探索法では，v と t の値を固定すると，

pがある値で探索回数がピークとなる．pの値が大きい場合，解の個数が多くて最初の初期値投入で解を捜し当ててしまう場合が多く，またpの値が小さいと，1つの節がカルノー図上で囲むセルの数が多くなるため，オーバーラップ数の傾斜がよりなだらかになっていくからである．上記の効率の上昇の少ない場合はpの値が大きい場合に相当し，実質的な性能向上は著しいといってよい．

3.2 局所探索法に不向きな例題

計算機実験によって，局所探索法はランダムに生成された例題に対してはとても効率が良いことがわかった（このことは他の報告[11]によっても確認されている）．そこで人為的に局所探索法にとって解きにくい例題生成法を考えた．このアルゴリズムでは，カルノー図上でオーバーラップ数の少ない方へと進んでいくのだから，解のまわりのセルのオーバーラップ数が多ければ，なかなか解にたどり着くことができなくなる．以下の生成アルゴリズムでは，ランダム性を保持し，項数や各項のリテラル数等幅広いパラメータ値を指定した上での生成が可能であるが，本稿では簡単のため，各項のリテラル数が3に限定されるいわゆる3SATの例題を生成する場合についてのみ述べる．リテラルの出現確率が小さい場合に相当し，2.2で述べたように，これは局所探索法にとって本来は得意な例題である．

3.2.1 アルゴリズム

生成アルゴリズム

ステップ1： ランダムに解X $(a_0, a_1, \cdots, a_{n-1})$を決める．

ステップ2： 変数$x_0, x_1, \cdots, x_{n-1}$の中からランダムに3つ選び，それらの肯定と否定によるリテラルの組合わせでできる8つの節を論理式に加える．

ステップ3： ステップ2で作った8つの節のうち，Xによって0になるものを取り除く．

ステップ4： 変数$x_0, x_1, \cdots, x_{n-1}$の中からランダムに3つ選び，$a_i$=0なら$x_i$とし，$a_i$=1なら$\overline{x_i}$とする．そして3つのリテラルのうちの1つをランダムに選び，それを反転させる．その3つリテラルによってできる節を論理式に加える．節が必要数に達するまでステップ4を繰り返す．

まずステップ2では，8個の節によって，カルノー図全体が覆われてしまうことを保証している．ところが，解を覆ってはいけないので，ステップ3でそのうち解を覆う節だけは除外することにしている．ステップ4では，まず3個のリテラルによって解を囲む節を生成し，それからそのうちの一つを反転させることによって，解に隣接する部分を囲む節にしている．ステップ4を行なうことによって，解の回りのセルが多くの節に囲まれていくことになり，解への到達を困難にしている．

3.2.2 解析

このアルゴリズムによって作られた例題のオーバーラップ数の傾斜の特徴を，調べてみる．ここでは，解からkの距離にある，

v	p	t	ランダム探索法	局所探索法
10	0.1	50	1096.43	6.08
10	0.3	50	473.83	9.89
10	0.5	50	16.09	2.97
10	0.7	50	1.86	1.50
10	0.9	50	1.13	1.11

表 1: 計算機実験の結果 (10 変数,50 節)

v	p	t	ランダム探索法	局所探索法
10	0.1	100	1200.22	5.80
10	0.3	100	1028.37	6.61
10	0.5	100	293.21	8.16
10	0.7	100	4.14	1.86
10	0.9	100	1.42	1.25

表 2: 計算機実験の結果 (10 変数,100 節)

v	p	t	ランダム探索法	局所探索法
10	0.1	200	975.51	5.82
10	0.3	200	971.22	6.19
10	0.5	200	845.94	13.28
10	0.7	200	14.21	3.17
10	0.9	200	1.75	1.41

表 3: 計算機実験の結果 (10 変数,200 節)

v	p	t	ランダム探索法	局所探索法
20	0.1	200	n.a.	11.00
20	0.2	200	> 20000	30.35
20	0.3	200	2259.99	18.14
20	0.4	200	9.64	2.43

表 4: 計算機実験の結果 (20 変数,200 節)

v	p	t	ランダム探索法	局所探索法
20	0.1	400	n.a.	11.00
20	0.2	400	n.a.	14.66
20	0.3	400	> 20000	70.51
20	0.4	400	91.22	4.59

表 5: 計算機実験の結果 (20 変数,400 節)

v	p	t	ランダム探索法	局所探索法
20	0.1	800	n.a.	10.84
20	0.2	800	n.a.	10.99
20	0.3	800	n.a.	32.93
20	0.4	800	> 20000	28.54
20	0.5	800	14.03	2.50

表 6: 計算機実験の結果 (20 変数,800 節)

ある1つのセルのオーバーラップ数を計算することで，解析をしてみる．

まず，解を$(0,0,\cdots,0)$としても一般性は失われない，これを囲むクローズは，$x_0+x_1+\cdots+x_{n-1}$である．次に解からkの距離にあるセルを

$$(1,1,\cdots,1,0,\cdots,0,0) \tag{1}$$

としても一般性は失われない．ただし，1の数はk個である．これを囲む節は，$\overline{x_0}+\overline{x_1}+\cdots\overline{x_{k-1}}+x_k+\cdots+x_{n-1}$である．上述のように解を定めた場合ステップ4で作られる節はランダムに3つの変数を選び，そのうちの1つを反転させたものであるから，この節が上にあげたセルを囲むための条件は以下の2つである．

- 3つの節のうち，x_0〜x_{k-1}の中から1つが選ばれ，x_k〜x_{n-1}の中から残り2つが選ばれる．
- ランダムに付けられる $\overline{\ }$ は，x_0〜x_{k-1}の中から選ばれたものである．

では，ステップ4に基づいて節を作ってみる．1番目の条件が当てはまる確率は，

$$\frac{{}_kC_1\cdot{}_{n-k}C_2}{{}_nC_3}=\frac{3k(n-k)(n-k-1)}{n(n-1)(n-2)} \tag{2}$$

2番目の条件として，この中でランダムに反転されるものがx_1〜x_kのうちの1つである確率は$\frac{1}{3}$である．

よって、(3.1)のセルがステップ4で作られる節によって囲まれる確率は，

$$\frac{k(n-k)(n-k-1)}{n(n-1)(n-2)} \tag{3}$$

になる．ここで，

$$f(k)=\frac{k(n-k)(n-k-1)}{n(n-1)(n-2)} \tag{4}$$

とおくと，$f(k)$は

$$k=\frac{2n-1-\sqrt{n^2-n+1}}{3}\simeq\frac{n}{3} \tag{5}$$

のとき極値をとる．これは極大値をとる値である．従って，選ばれた初期値の解からの距離が，$\frac{n}{3}$より小さければ，平均的に，その初期値から解へとたどりつける．全体でセルは2^n個あるので，初期値がうまく選ばれる確率は，

$$\frac{{}_nC_0+{}_nC_1+\cdots+{}_nC_{\frac{n}{3}}}{2^n}\leq e^{-\frac{n}{36}} \tag{6}$$

となる．これは非常に小さい値であるため，この例題生成法による例題に対しては，局所探索法は，あまり良い性能を発揮できないものと思われる．

しかし，失敗した時は解の全ての変数の値を反転させたセルの近くにたどりつく可能性が非常に高いことに注目されたい．従って，失敗した時は最後にたどりついたセルの全ての変数の値を反転させたところに次の初期値を選んで再度探索を行なえば，短時間で解にたどりつける可能性が高い．

3.3 ハミルトン閉路問題のSATへの変換例題

次に，ランダム性の少ない，より実際的な例題に対して評価してみる．“実際的な例題”がどのようなものであるかについては議論があろうが，ここではある問題をSATに変換した例題をとり上げてみる．ランダム生成はすべての例題をある程度まんべんなく生成するのに対し，本手法による例題生成には明らかに偏りが存在する．ここではNP完全問題の1つとして知られているハミルトン閉路問題をSATに変換した例題を用いることにする．ハミルトン閉路問題とは，グラフ$G=(V,E)$(Vは頂点の集合，Eは辺の集合)が与えられたとき，グラフGの全ての頂点を1回ずつ通る閉路(ハミルトン閉路)が存在するかどうかを問う問題である．

このような議論に当っては，どのような変換アルゴリズムを用いるかが鍵となってくる．多項式時間変換（PTR）でなくては意味がないことは明らかであるが，PTRであればどんなものでも良いという訳にはいかず，NP完全性を証明する場合のような大雑把な議論では明らかに不十分である．以下では2種類のPTRを提案する．第一のTPRはN頂点M枝のグラフをNM変数のCNF式に変換するものである．第二のPTRは$N(\frac{1}{k}logN+\frac{k-1}{k}logD)$変数のCNF式への変換である．ここで$D$はグラフの最大次数であり，$k$は定数（実用的には3程度まで）である．この第二のPTRは変数の数に関しては最適に近いものと思われる（その最適性の証明は重要な課題である）．なお，節の数については本稿の範囲では考慮の対象から外した．局所探索法が節の数が（ある一定以上）多くなったときに逆に効率が上がるという性質を持っているからである．

3.3.1 NM変数による変換

ハミルトン閉路問題のYESの例題の特徴として，次の二つが考えられる．

1. 全ての頂点を通る閉路が存在する．

2. 閉路は1つしか存在しない．

より直観的には，1は各頂点から枝を2本ずつ選ぶことができるということである．また2では，選ばれた枝に番号を順番につけていくと，N番目の枝と1番目の枝が同じ頂点から出ているということである（Nは頂点の数）．より具体的には，ある頂点から選ばれた枝2本に1とNという番号をつけてやり，各頂点においては選ばれた2本につけられる番号は続き番号でなければいけないようにする．このことによって，閉路が1つしかない場合はうまく1週してN番目の枝まで来るが，閉路が2つ以上ある場合にはどこかで破綻を来すことになる．

そこで，上述の1と2を一緒にしてx_{ij}という変数を使うことにする．これは枝jがi番目の枝として選ばれたとき1となりそれ以外では0になるというふうに考える．i番目の枝というものはただ1つ存在しなければならないのだから，x_{ij}，は各iに対して1つだけ1である．また，ある頂点から伸びている枝の数をlとすると，その頂点について対象となる変数はNl個あるのだが，これらのうち2つが1で残りはすべて0でなければならない．し

かも，1となるべき2つの変数のiの値は続いてなければならない．また頂点1に関しては，選ばれた枝の順番は1番目とN番目である．

変換アルゴリズム　与えられたハミルトン閉路問題のグラフの頂点の数をN，枝の数をMとする．

ステップ1： $1 \leq i \leq N$に対して，$(x_{i1}+x_{i2}+\cdots+x_{iM})$をつくる．

ステップ2： $1 \leq i \leq N$に対して，$x_{i1}, x_{i2}, \cdots, x_{iM}$の中から2つを選びそれを$x_{ia}, x_{ib}$とすると，全ての$a, b$の組合せに対して，$(\overline{x_{ia}}+\overline{x_{ib}})$をつくる．

ステップ3： 頂点1から伸びている枝を$(11),(12)\cdots(1s)$として，$(x_{1(11)}+x_{1(12)}+\cdots+x_{1(1s)})(x_{N(11)}+x_{N(12)}+\cdots+x_{N(1s)})$をつくる．

ステップ4： 頂点$h(\neq 1)$に対して，hから出ている枝を$(h1),(h2),\cdots,(hs)$とし，$(x_{1(h1)}+x_{2(h1)}+\cdots+x_{N(h1)}+x_{1(h2)}+x_{2(h2)}+\cdots+x_{N(h2)}+\cdots+x_{1(hs)}+x_{2(hs)}+\cdots+x_{N(hs)})$をつくる．

ステップ5： ステップ4に出てくるNs個の変数のうち1つを選び，それをx_pとしてすべてのpに対してステップ4の節と同じで，x_pだけに$\bar{}$をつけたものをつくる．

ステップ6： ステップ4に出てくるNs個の変数のうち3つを選び，それらをx_a, x_b, x_cとしてすべてのx_a, x_b, x_cの組合せに対して，$(\overline{x_a}+\overline{x_b}+\overline{x_c})$を作る．

ステップ7： ステップ4に出てくるNs個の変数のうち1つを$x_{i(ha)}(1 \leq i \leq M-1, 1 \leq a \leq s)$とする．これに対して$x_{j(hb)}(i+2 \leq j \leq N, b \neq a)$を選んで，$a, b$の全ての組合せに対して$(\overline{x_{i(ha)}}+\overline{x_{j(hb)}})$をつくる．

まず，ステップ1とステップ2でx_{ij}の各iに対して1になる変数が1つだけということを保証している．ステップ1で全ての変数が0になることがないようにし，ステップ2で2つ以上の変数が1にならないようにしている．次に，ステップ3では頂点1から異なった2本の枝が，1番目とN番目として選ばれなければならないことを示している．最後にステップ4～7では各頂点から枝が2本ずつ続き番号で選ばれなければならないことを表わしている．

計算機実験　計算機実験は，上述のように変換した例題と，それに対して変数と節の数が同じであるランダムに生成された例題に対して行なった．結果は表7と表8に示す通りである．

この結果を見てみると，元のハミルトン閉路問題では頂点数と枝数が少ないものに対しても，かなりの時間がかかっていることがわかる．この変換によると変数の数がかなり増えてしまうためであると思われる．また，変数とクローズの値が同じ数のランダムに生成された例題と比較しても，はるかにハミルトン閉路問題の例題の方が解きにくいことがわかる．これは，局所探索法が，例題の節中に含まれるリテラルの数が少ないときにより性能を発揮するのに対し，この変換方法によると，様々な大きさの節が入り混じっているためであると推測される．

頂点数	枝数	変数	節数	探索回数
3	5	15	138	28.64
4	6	24	507	82.61
4	8	32	1317	166.81

表7: 計算機実験（ハミルトン閉路問題の例題）

v(変数)	p	l(節数)	探索回数
15	0.1	138	8.52
15	0.3	138	20.74
15	0.5	138	2.34
15	0.7	138	1.23
24	0.1	507	13.35
24	0.3	507	26.64
24	0.5	507	1.44
24	0.7	507	1.01
32	0.1	1317	16.54
32	0.3	1317	7.77
32	0.5	1317	1.12
32	0.7	1317	1.00

表8: 計算機実験（ランダム生成の例題）

3.3.2　$N(\frac{1}{k}logN+\frac{k-1}{k}logD)$変数による変換

前述の変換方法では，SATにする時の変数が多くて効率の悪いものとなってしまった．そこで少ない変数で変換できる方法を考えた．この方法では（ハミルトン閉路をたどることを意識して）各頂点に番号を1からNまで重複なくつけていく．また頂点どうしが枝でつながれていないときには，その2つの頂点には連続した番号をつけてはいけないことにする．変数を少なくするために頂点につける番号は2進にする．（以下説明を簡単にするためにk=2の場合を示す．"奇数"をkの倍数，"偶数"をkの倍数以外と置き換えれば一般のkの場合になる．）そして付された番号が奇数だった頂点に関しては直接頂点番号を表し，偶数だった頂点に関しては1つ前の番号の頂点からでた何番目の枝に継っている頂点であるという表し方にする．変数は$x_{ij}(2 \leq i \leq N, i$は偶数$, 1 \leq j \leq g, g$は$logD$の小数点以下を切り上げた整数$)$と$y_{ij}(2 \leq i \leq N, i$は奇数$, 1 \leq j \leq p, p$は$logN$の小数点以下を切り上げた整数でNが2の乗数だったら1加える$)$を用意する．例えば$(y_{51}, y_{52}, y_{53}, \cdots, y_{5p})=(1,1,0,\cdots,0), (x_{61}, x_{62}, x_{63}, \cdots, x_{6g})=(0,1,0,\cdots,0)$だった場合には5番目の頂点として頂点3が，6番目の頂点として頂点3から出ている枝の2番目に継っている頂点が選ばれたことを意味する．ここでは枝の順序づけとして，枝番号の小さい順に0からつけていくことにする．頂点1には番号1を固定することにして，頂点2からNに番号2からNまでをつけていくことにする．

変換アルゴリズム　与えられたハミルトン閉路問題のグラフの頂点の数をN，枝の数をMとする．まず，定数として$b_{ij}(2 \leq i \leq$

$N, 1 \leq j \leq p$) と $a_{ij}(0 \leq i \leq D-1, 1 \leq j \leq g)$ を用意する．$b_{i1}, b_{i2}, \cdots, b_{ip}$ は2進で i を表すものである．頂点 K の次数を d_K とし，頂点 K から出ている枝を枝番号が小さい順番に $K_0, K_1, \cdots, K_{d_K}$ とする．

ステップ1： $2 \leq i \leq N$(i は奇数) という各 i に対して $(y_{i2} + y_{i3} + \cdots + y_{ip})$ をつくる．

ステップ2： $2 \leq i \leq N$(i は奇数) という各 i に対して以下の操作を行なう．

$1 \leq j \leq p$ の各 j に対して b_{Nj} を見て，b_{Nj}=0 ならば節を1つつくる．つくり方は以下の通りである．

$\overline{y_{ij}}$ を節中に加える．$j+1 \leq k \leq p$ に対して b_{Nk} を見て，b_{Nk}=1 なら節中に $\overline{y_{ik}}$ を加える．

ステップ3： $1 \leq i \leq N, 2 \leq k \leq N$($i$ は奇数) となる各 i,k に対して次の操作を行なう．

$1 \leq s \leq g$ に対して a_{d_k-1s} を見て，a_{d_k-1s}=0 だったら節を1つつくる．つくり方は以下の通りである．

$\overline{x_{i+1s}}$ を節中に加える．$s+1 \leq l \leq g$ に対して a_{d_k-1l} を見て，a_{d_k-1l}=1 ならば $\overline{x_{i+1l}}$ を節中に加える．$1 \leq h \leq p$ に対して a_{j-1l} を見て，$b_{kh}=1$ ならば $\overline{y_{ih}}$ を，$b_{kh}=0$ ならば y_{ih} を節中に加える (i=1 に対しては行なわない)．

ステップ4： $2 \leq i < j \leq N, 2 \leq k \leq N$($i,j$ はともに奇数) という i,j,k の各組合せに対して節を1つつくる．つくり方は以下の通りである．

$1 \leq s \leq p$ の s に対して b_{ks}=1 ならば $\overline{y_{is}}$ と $\overline{y_{js}}$ を b_{ks}=0 ならば y_{is} と y_{js} を節中に加える．

ステップ5： $1 \leq i \leq N, 2 \leq j \leq N$($i,j$ は奇数) という i,j に対して，以下の操作を行なう．

$2 \leq k \leq N, o \leq s \leq d_k - 1$ の各 k,s に対して節を1つつくる．つくり方は以下の通りである．ただし，枝 k_s は頂点 k と k' に継っているものとする．

$1 \leq h \leq p$ に対して，b_{kh}=1 ならば $\overline{y_{ih}}$ を，b_{kh}=0 ならば y_{ih} を節中に加える (i=1 に対しては行なわない)．$1 \leq h \leq g$ に対して，a_{sh}=1 ならば $\overline{x_{i+1h}}$ を，a_{sh}=0 ならば x_{i+1h} を節中に加える．$1 \leq h \leq p$ に対して，$b_{k'h}$=1 ならば $\overline{y_{jh}}$ を，$b_{k'h}$=0 ならば y_{jh} を節中に加える．

ステップ6： $1 \leq i \leq N, 2 \leq j \leq N$($i,j$ は奇数) という全ての i,j の組に対して $2 \leq k \leq N, 2 \leq k' \leq N, 0 \leq s \leq d_k - 1, 0 \leq s' \leq d_{k'} - 1$ という組合せで，k_s の k でない方の頂点と $k'_{s'}$ の k' でない方の頂点が一致した場合には次のクローズをつくる．

$1 \leq h \leq p$ に対して b_{kh}=1 なら $\overline{y_{ih}}$ を，b_{kh}=0 なら y_{ih} を節中に加える．(i=1 のときは行なわない．) $1 \leq h \leq g$ に対して a_{sh}=1 なら $\overline{x_{i+1h}}$ を，a_{sh}=0 なら x_{i+1h} を節中に加える．$1 \leq h \leq p$ に対して $b_{k'h}$=1 なら $\overline{y_{jh}}$ を，$b_{k'h}$=0 なら y_{jh} を節中に加える．$1 \leq h \leq g$ に対して $a_{s'h}$=1 なら $\overline{x_{j+1h}}$ を，$a_{s'h}$=0 なら x_{j+1h} を節中に加える．

ステップ7： $2 \leq i \leq N-2$(i は奇数) に対して以下の操作を行なう．

$2 \leq k \leq N, 0 \leq s \leq d_k - 1$ に対して，k_s の k と継っていない方の頂点を k' とする．k' と継っていない頂点を k'' とすると，全ての k'' に対して以下の節をつくる．

$1 \leq h \leq p$ に対して，b_{kh}=1 ならば $\overline{y_{ih}}$ を，b_{kh}=0 ならば y_{ih} を節中に加える．$1 \leq h \leq g$ に対して a_{sh}=1 なら $\overline{x_{i+1h}}$ を，a_{sh}=0 なら x_{i+1h} を節中に加える．$1 \leq h \leq p$ に対して $b_{k'h}$=1 なら $\overline{y_{i+2h}}$ を，$b_{k'h}$=0 なら y_{j+2h} を節中に加える．

ステップ8： $0 \leq s \leq d_1 - 1$ に対して，枝 1_s と継っている，1でない方の頂点を k とする．k と継っていない全ての頂点 k' に対して以下の節をつくる．

$1 \leq h \leq g$ に対して，a_{sh}=1 なら $\overline{x_{2h}}$ を，a_{sh}=0 なら x_{2h} を節中に加える．$1 \leq h \leq p$ に対して $b_{k'h}$=1 なら $\overline{y_{3h}}$ を，$b_{k'h}$=0 なら y_{3h} を節中に加える．

ステップ9-1(N が偶数の場合)： $2 \leq k \leq N, 0 \leq s \leq d_k - 1$ に対して，枝 k_s が継っている，k ではない方の頂点を k' とする．頂点 k' と頂点1が継っていなかったら以下の節をつくる．

$1 \leq h \leq p$ に対して，b_{kh}=1 ならば $\overline{y_{n-1h}}$ を，b_{kh}=0 ならば y_{n-1h} を節中に加える．$1 \leq h \leq g$ に対して，a_{sh}=1 なら $\overline{x_{nh}}$ を，a_{sh}=0 なら x_{nh} を節中に加える．

ステップ9-2(N が奇数の場合)： $2 \leq k \leq N$ に対して，頂点 k が頂点1継っていなかったら次の節をつくる．

$1 \leq h \leq p$ に対して，b_{kh}=1 ならば $\overline{y_{nh}}$ を，b_{kh}=0 ならば y_{nh} を節中に加える．

ステップ1では，$y_{i1}, y_{i2}, \cdots, y_{ip}$ が2進の0と1を表したとき0になるような節をつくっている．ステップ2では $y_{i1}, y_{i2}, \cdots, y_{ip}$ が N を越えたとき0になる節をつくっている．ステップ3では $x_{i1}, x_{i2}, \cdots, x_{ig}$ がその前の頂点の次数を超えたら0になる節をつくっている．ステップ4～6では頂点が重複して選ばれないようにしている．ステップ4では奇数番目と奇数番目，ステップ5では奇数と偶数，ステップ6では偶数と偶数についてそれぞれ見ている．ステップ7～9では偶数番目から奇数番目の連続する頂点には枝がなくてはいけないということを表している．ステップ8では1番目の頂点について，ステップ9では N 番目の頂点についてそれぞれ特別な処理をしている．

計算機実験　計算機実験は NM 変数の実験と同じ頂点数，枝数の例題と，頂点数や枝数を増やしたものに対して行なった．これは、上述の D が N の場合に対して行なったものである．また，変数の数と節の数が同じであるランダム生成例題に対しても実験を行なって比較した．実験結果を表9及び表10に示す．この結果から分かるように，ハミルトン閉路問題を変換した例題の探索回

頂点数	枝数	変数	節数	探索回数
3	5	4	4	2.78
4	6	9	23	11.58
4	8	9	22	11.23
10	20	36	435	89.24
10	30	36	435	94.05
10	40	36	420	73.47
10	50	36	407	66.66
10	60	36	397	80.31
10	70	36	389	66.83
10	80	36	381	73.89
15	100	56	1446.56	149.02
15	150	56	1390.78	160.89
15	200	56	1352.68	177.13

表 9: 計算機実験（ハミルトン閉路問題の例題）

v(変数)	p	t(節数)	探索回数
36	0.1	400	40.58
36	0.3	400	1.94
36	0.5	400	1.00
36	0.7	400	1.00
56	0.1	1400	299.49
56	0.3	1400	1.17
56	0.5	1400	1.00
56	0.7	1400	1.00

表 10: 計算機実験（ランダム生成の例題）

数はランダム生成例題の最悪の場合に比べれば悪くない．もちろん NM 変数による変換の場合に比べて扱えるグラフの頂点数は大幅に上昇している．

4. おわりに

本研究では，中心となる局所探索法をいろいろな角度から評価することを目標としてきた．まず，局所探索法を C 言語で記述し，ランダム生成例題に対する計算機実験を行なった．これは純粋なランダム探索法に対しても同じパラメータの例題を与えて，その探索回数を比較したものである．この結果，局所探索法ははるかに効率の良いアルゴリズムであることが分かった．ただ，このアルゴリズムは YES の例題に対して充足解を見つけることには優れているが，YES か NO かの判定を下せないところに問題点が残る．次に，局所探索法はどのような例題集合に対しても効率が良いのだろうか，との疑問を抱き，局所探索法に不向きな例題の生成法を考えた．この例題に対する局所探索法の効率を調べるために数学的な解析を行なった結果，確かに不向きな例題であることが分かった．最後に，ハミルトン閉路問題の例題を SAT に変換して局所探索法に与えてみた．変換は 2 種類考えた．あたりまえの結果ではあるが，少ない変数で変換した方が探索回数が大幅に減少することが分かった．このことからも分かるように，1 つの例題を SAT に変換する方法は非常に多く存在する．また，局所探索法の探索回数は変数の数に大きく関係している．従って，ある NP 完全問題を SAT に変換するときに必要な変数の数の下限を，その問題の難しさの指標として見ることは，合理的かも知れない．

今後の課題としては，アルゴリズム局所探索法が SAT の本来の目的である YES，NO の判定ができるように改良してみたい．また，いろいろな NP 完全問題を SAT に変換する方法を考えると共に，それによってつくられた例題がどのような偏りを持ったものかを調べ，局所探索法への向き不向きを解析等によって調べていきたい．

参考文献

[1] A. Goldberg, P. Purdom and C. Brown. Average time analyses of simplified Davis-Putnam procedures. *Inform. Process. Lett.*, pp72-75,1982.

[2] K. Iwama. CNF satisfiability test by counting and polynomial average time. *SIAM J.Comput.*, pp.385-391,1989.

[3] K. Iwama, H. Abeta and E. Miyano. Random generation of satisfiable and unsatisfiable CNF predicates. In *Proc. 12th IFIP World Computer Congress*, pp.322-328,1992.

[4] K. Bugrara, P. Purdom. Average time analysis of clause order backtracking. *Indiana University Technical Report.*, 1990.

[5] E. Koutsoupias and C. Papadimitriou. On the greedy algorithm for satisfiability. *Information Processing Letters 43*, pp.53-55,1992.

[6] D. Mitchell, B. Selman and H. Levesque. Hard and easy distributions of SAT problems. In *Proc. Tenth National Conference on Artificial Intellegence.* pp.459-465,1992.

[7] 宮崎，岩間．CNF 論理式の充足解に対する改良されたランダム探索法．平成 4 年度九州支部連合大会（平成 4 年 10 月）．

[8] P. Purdom and C. Brown. An analysis of backtracking with search rearrangement. *SIMA J.Comput.*,pp.717-733,1983.

[9] P. Purdom and C. Brown. The pure literal rule and polynomial average time. *SIMA J.Comput.*, pp.943-953,1985.

[10] P.Purdom. A survey of average time analyses of satisfiability algorithms. *Journal of Information Processing*, Vol.13,No4, pp.449-455,1990.

[11] B. Selman, H.Levesque and D. Mitchell. A new method for solving hard satisfiability problems. *AAAI, San Jose, CA*, pp.440-446,1992.