A-061

論理式の解密度の濃縮
Density Condensation of Boolean Formulas

花谷 陽一　　堀山 貴史　　岩間 一雄
Graduate School of Informatics, Kyoto University

1. はじめに

n 変数の論理式 f が与えられた時に、充足可能性を保存したまま他に “良い性質” を持たせた論理式 g を生成したいことが多くの場面で見受けられる。例えば、実世界の問題に基づいた論理式を 1 つベンチマークとして持っている時に、これをもとにして他の類似の論理式を生成できれば望ましいだろう。また、与えられた論理式を排他的論理和を使わない形に変形したいなどの要求もあるだろう。これらの中で、f の充足解の得やすさもまた、良い性質として自然であろう。

変数を少なくすれば常に充足解が得やすくなるわけではなく、充足解の密度すなわち (充足解の個数)/2^n を高めれば解が得やすいという主張の方が理にかなっている。そこで、解の濃縮問題 (DCP) を、論理式 f が与えられた時に以下を満たす g を求める問題と定義する: (1) g は f の充足可能性を保存する。(2) f が充足可能ならば、g の解の密度は f よりも大きい。ここで、f の充足可能性判定はこの特殊な場合 (条件 (2) で密度を 1.0 にすることが求められる) であることに注意が必要である。この “完全な濃縮” は NP 完全であることが知られているので、DCP はその一般化として重要な役割を担っている。

本稿では、DCP を解く 3 種のアルゴリズムを示し、その性能を比較する。簡単のため、入力を 3-CNF に限定する。出力については、以下の 3 種の制限が挙げられるだろう。(1) CNF のみ。(2) 式の形には制限を設けないが、論理演算のみを用いる。(3) 関数代入を許す。(1) の場合には、これを繰り返すことで解の濃度を任意に高めることができるため、DCP を解くのは困難であろう。(3) はあまり困難ではなく、密度を指数的に高める多項式時間アルゴリズムが存在する。したがって、(2) が最も興味深い問題となる。

本稿では、SAT アルゴリズムに基づき、それをシミュレートする論理式を生成する手法をとる。また、アルゴリズムの性能評価のために、指標として密度 d および比 $|g|/d$ を用いる。

2. 解密度の濃縮アルゴリズム

2.1 ランダムフリップに基づく方法

次のようなアルゴリズムを考える。
(1) 初期割り当て $a \in \{0,1\}^n$ をランダムに選択する。
(2) ランダムに選択された変数の値をフリップする。
(3) ステップ (2) を t 回繰り返す。繰り返しの中で、充足する割り当てを得れば True を出力し、そうでなければ False を出力する。

上記のアルゴリズムをシミュレートすることにより、論理式 g を構築する。初期割り当てを $x^0 = (x_1^0, \ldots, x_n^0)$ と表し、k 番目のフリップを行ったときの割り当てを $x^k = (x_1^k, \ldots, x_n^k)$ と表す。また、$v^k = (v_1^k, \ldots, v_{\lceil \log n \rceil}^k)$ を用いて k 番目のフリップでランダムに選ばれる変数を表す。ここで、v^k は 2 進数の並びであるとみなして、変数 x_{v^k} が選択されたことを表す。以下に示す式によってアルゴリズムをシミュレートすることができる。

$$\bigvee_{k=0}^{t} f(x_1^k, \ldots, x_n^k) \tag{1}$$

k ステップ目の割り当て $(x_1^k, \ldots, x_n^k)$ は、$x^0, v^1, \ldots, v^k$ で表すことができる。

$$x_i^k = x_i^0 \oplus (v^1 = i) \oplus \cdots \oplus (v^k = i) \tag{2}$$

式 (1) の変数 x_i^k を式 (2) を用いることで、変数 $x^0, v^1, \ldots, v^k$ で表される論理式 g を得る。

g の式の長さとその充足解の密度 d を見積もる。式 (2) より、x_i^k の長さ $|x_i^k| = 1 + k\lceil \log n \rceil$ であるから、g の式の長さは、

$$|g| = \sum_{k=0}^{t} |f| \cdot (1 + k\lceil \log n \rceil) \sim |f| t^2 \log n$$

となる。g はランダムフリップアルゴリズムを記述しているので、g の充足割り当ての密度 d は、f の充足割り当て a^* を得る確率と一致する。初期割り当て a が、充足割り当て a^* からのハミング距離 k であるとき、k 回のフリップで a^* にたどり着く確率は $k!/n^k$ であるから、密度 $d = \left(\frac{1}{2}\right)^n \sum_{k=0}^{t} \binom{n}{k} \frac{k!}{n^k} \sim \left(\frac{1}{2}\right)^n$ を得る。

2.2 確率的局所探索に基づく方法

Schöning の確率的局所探索アルゴリズム [2] は以下の通りである。
(1) 初期割り当て $a \in \{0,1\}^n$ をランダムに選択する。
(2) 非充足な項の中から任意の項を選び、それに含まれる変数のひとつをフリップする。
(3) ステップ (2) を $3n$ 回繰り返す。

このアルゴリズムの中のフリップする変数の選択について、我々は非充足な項をランダムに選ぶのではなく、充足しない初めの項 C_i を選択する。これは、$C_1, C_2, \ldots, C_{i-1}$ は充足していることを意味する。このような改変を加えても、充足割り当て a^* を得る確率は保証されている。なぜなら、各ステップで選択の中で少なくとも 1 通りは充足割り当てとのハミング距離を縮めるからである。

我々は、この改変された Schöning のアルゴリズムをシミュレートすることで、g を構築する。

初期割り当てを $x^0 = (x_1^0, \ldots, x_n^0)$ と表し、k 番目のフリップを行ったときの割り当てを $x^k = (x_1^k, \ldots, x_n^k)$ と表す。また、(y^k, z^k) を用いて k 番目のフリップで項の中からランダムに選ばれる変数を表す。ここで、$(y^k, z^k) = (0,1)((1,0),(1,1))$ は、選ばれた項の中の 1 番目 (2, 3 番目) の変数がフリップされることを表す。したがって、k

ステップ後の割り当て $(x_1^{k+1},\ldots,x_n^{k+1})$ を以下のように得る。

$$x_i^{k+1} = x_i^k \oplus \bigvee_{l\text{s.t.}x_i\in C_l} \left(\bigwedge_{1\le j<l} \overline{C_j^k} \right) C_l^k \mathrm{pos}_{i,l}(y^k, z^k) \quad (3)$$

ここで、$\mathrm{pos}_{i,l}(y^k, z^k)$ は、x_i が項 C_l の 1 番目 (2, 3 番目) に出現するとき、$\overline{y^k}\cdot z^k(y^k\cdot\overline{z^k}, \overline{y^k}\cdot\overline{z^k})$ となる。また、C_{i_k} は、$C_i = x_{i_1}\vee x_{i_2}\vee x_{i_3}$ として、$x_{i_1}^k \vee x_{i_2}^k \vee x_{i_3}^k$ を表す。

式 (3) を用いて変数を消去すると、x_i^{k+1} は $(x_1^0,\ldots,x_n^0),(y^0,z^0),\ldots,(y^k,z^k)$ の関数と見ることができる。したがって、$f(x_1^t,\ldots,x_n^t)$ は $n+2t$ 変数の関数 g と与えることができる。

g の式の長さとその充足解の密度 d を見積もる。漸化式 (3) によって、x_i^{k+1} の長さは、f の項数を m として、$|x_i^{k+1}| < |x_i^k| + (3\max_i|x_i^k|+2)m$ となるので、$|x_i^t| \sim (3m)^t \max_i |x_i^0|$ となる。これは、g の長さが $|g|\sim(3m)^t|f|$ であることを示す。

我々は、(y^k,z^k) を用いて各ステップのランダムな選択を決定し、少なくとも確率 $1/4$ で a^* とのハミング距離を 1 近付けるので、g の解の密度は、Shcöning のアルゴリズムで f から解を得る確率と等しくなり、

$$d = \left(\frac{1}{2}\right)^n \sum_{j=0}^{t} \binom{n}{j}\left(\frac{1}{4}\right)^j > \frac{1}{2^n}\cdot\frac{n^t}{t!\cdot 4^t} \sim \frac{1}{2^n}\left(\frac{en}{4t}\right)^t$$

を得る。

2.3 探索空間分割に基づく方法

我々は以下のような分割手続き split(F) を用いる。

procedure split(F)
(1) F が 2-CNF ならば、多項式時間で F を解く。
(2) F から項 $x_i\vee x_j\vee x_k$ を選択する。
(3) 以下の式を返す。

$$\mathrm{split}(F|_{x_i=1})\vee\mathrm{split}(F|_{x_i=0,x_j=1})\vee\mathrm{split}(F|_{x_i=0,x_j=0,x_k=1}).$$

この手続き split を t 個の変数が消去される深さまで適用する。このようにして、代入が途中まで行われた式 (途中式) が論理和で結合された形の式 g を得る。f が充足可能ならば、g 中の途中式が少なくとも一つは充足可能であることに注意
されたい。$t>1$ になると、複数の途中式で異なった変数が消去されるが、変数のラベルの付け替えを行えば g は $n-t$ 変数の論理式となる。

g の式の長さは、$|g|\sim|f|\cdot 1.839^t$ となる。また、g の充足解の密度 d は、$(1/2)^{n-t}$ となる。

3. アルゴリズムの評価

我々は、論理式の濃縮アルゴリズムの評価に 2 通りの方法を用いる。ひとつは、解の密度 d である。これは、ごく自然な評価方法である。もうひとつは、$|g|/d$ である。これは、ランダムな割り当てによって期待値 $1/d$ で g の充足解を見つけることができ、一つの割り当ての計算が $|g|$ であることに拠る。より大きな d、より小さな $|g|/d$ を得るアルゴリズムが望ましい。アルゴリズムの性能比較のため、尺度 d, $|g|/d$ を g の計算に要する時間 $T(=|g|)$ の関数として漸近的に求め、表 1 に示す。これより、計算時間に対する各アルゴリズムの振る舞いは図 1, 2 のようになる。点線がランダムフリップ、細線が確率的局所探索、太線が空間分割のアルゴリズムである。

表 1: 提案アルゴリズムに関する T の比較

	解の密度 d	$\lvert g\rvert/d$
ランダムフリップ	定数	T
確率的局所探索	$T^{\varphi(n)}$	$T^{\chi(n)}$
空間分割型	$T^{1.137}$	$T^{-0.137}$

$\varphi(n)=\frac{1}{\log 3n^3}\log\frac{4n\log 3n^3}{e\log T}$, $\chi(n)=\frac{1}{\log 3n^3}\log\frac{12n^2\log T}{e\log 3n^3}$,

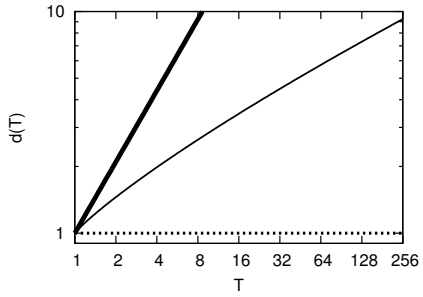

図 1: アルゴリズムがかけた時間に対する d の変化

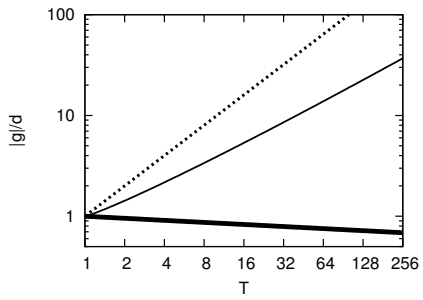

図 2: アルゴリズムがかけた時間に対する $|g|/d$ の変化

これら 3 つのアルゴリズムは、計算時間をかけることで解の密度を任意に増加させることができるが、式の長さは長くなる。ランダムフリップのアルゴリズムは、密度の増加に比べて式の長さの伸びが速い。すなわち、何も戦略を持たないフリップでは、良好な濃縮が得られない。Schöning のフリップは、密度を指数的に増加させる。しかし、出力する式の長さが $(3n^3)^k$ となり、d の増加よりも遥かに速い。図 1, 2 より、空間分割型のみが d と $|g|/d$ という両方の尺度で良い結果を示している。

参考文献

[1] B. Monien and E. Speckenmeyer, “3-Satisfiability is Testable in $O(1.62^r)$ Steps”, Technical Report Bericht Nr. 3/1979, Universität-Gesamthochschule-Paderborn, 1979. Reihe Theoretische Informatik.

[2] U. Schöning, Constraint Satisfaction Problems”, Proc. 40th FOCS, pp.410–414, 1999.