ＢＤＤの制約順序の効率化による制約充足問題の解法

1D－1

石田　賢三　斎藤　逸郎　石塚　満
東京大学工学部電子情報工学科石塚研究室
奥乃 博 (NTT 基礎研究所)

１、はじめに

二分決定グラフ（ＢＤＤ）を使って組合せ問題の全解を求めることを検討する。ここでの課題に、計算途中に起こりうるノード数の爆発を避け、同じ計算機資源のもとでできるだけ大きな問題が解けるようにする方法を開発することがあげられる。組合せ問題にＢＤＤを適用する場合には最大ノード数が重要であること、およびそれが変数順序だけではなく制約組合せ順序の影響を受けることがわかっている［１］。本稿では、制約組合せ順序を求める方法の一つを提案し、具体的に有効な場合と無効な場合を実験検証してみる。

２、二分決定グラフ（ＢＤＤ）

ＢＤＤは、Akersが提案し、Bryantが考案した論理関数のグラフによる表現法である。一般に論理関数は変数のとる値によって木構造で表現できる。この木表現に変数順序を導入し、かつ、終端ノード、重複ノードを共有化し、冗長ノードを削除することによって既約化した木構造を既約順序付き二分決定グラフ（Reduced Ordered BDD,ROBDD）と呼ぶ。ROBDDは、さまざまな属性を有した枝を導入し、ノード数を減らす工夫がされている。ROBDDを論理関数間で共有できるようにした木構造を共有二分決定グラフ（Shared BDD,SBDD）と呼ぶことにする。

ＢＤＤの主な特徴は

（１）　変数順序を固定すると論理関数を一意に表すことができる。

（２）　多くの実用的な関数を比較的コンパクトなノード数で表現できる。

（３）　論理演算に対する演算がノード数に比例する時間で行える。

とまとめることができる。これらの特徴によって、ＢＤＤは論理設計や検証、テスト生成など多岐にわたる応用で使用されている。

３、算術論理式計算システムＢＥＭ－II

本稿では、ＢＤＤシステムとして算術論理式システムＢＥＭ－IIを使用した。ＢＥＭ－IIはＳＢＤＤをもとにしたシステムであり、入力として論理関数だけでなく、算術演算をも含む論理式が許される。

４、変数順序と制約順序

ＢＤＤの計算途中で必要なノード数は変数順序だけでなく、ＢＤＤ間の制約組合せの順序によっても大きく変化する。組合せ問題のコーディングからそのままＢＤＤを生成すると容易に組合せの爆発が生ずるとき、問題に応じた制約順序を求めることが重要である。

５、制約組合せ条件の順序付け

本節では、包含に着目した制約順序を決めるアルゴリズムを提案する。まず、制約順序を求めるための基本方針を示す。それは以下のようにまとめることができる。

・多くの制約条件で使われる変数はノード数を減少させやすいので、よく使われる変数を包含する制約条件を優先する。

A Solution of Constant Satisfaction Problems by Using an Effective Constant Ordering of BDD

Kenzo Ishida,Itsuro Saito,Misturu Ishizuka,Hiroshi Okuno
The University of Tokyo
7-3-1 Hongo,Bunkyo-ku,Tokyo 113,Japan

・すでに実行された制約条件と包含関係にある制約条件のうち、変数の少ないものを優先する。

次に制約順序を求めるアルゴリズムを示す。

（１）各変数ごとに制約条件に使われる回数を調べ、最大値をhに代入する。

（２）制約条件で使われる回数がh以上である変数のみで構成される制約条件の中で変数の少ない条件を選択する。
そのような制約条件がない場合には
h=h-1
として（２）を繰り返す。

（３）選択された制約条件群に包含される制約条件を変数の少ない順に制約条件として格納する。そして、（２）に戻る。

６、実験結果

３次の魔法陣を取り上げて、提案したアルゴリズムによって決定された制約順序を評価する。

さまざまな変数順序によって、最終ノード数と計算途中で必要となる最大ノード数がどう変わるかを実験した。比較の対象としてほぼ最適な制約条件と変数順序の組を求めるアルゴリズムＣＣＶＯ（Correlation-based Constraint and Variable Ordering）による制約順序を利用した。

また変数順序は論文［１］による変数順序を利用した。

これによる実験結果は図１および図２のようになる。本アルゴリズムは変数順序の影響が比較的少なかった。

７、本アルゴリズムの評価

本アルゴリズムは、多くの制約条件で使われている変数を手がかりに制約条件を決定するので使用される回数に差があまりみられない場合には、思うようにノードの減少が見られない。（実際、８－Ｑｕｅｅｎｓにおいて適応したがノード数の減少は見られなかった。）しかしアルゴリズムが比較的簡単でプログラム化し易いので、大規模なコーディングにおいても制約条件の並び替えが容易に行える。具体的に全加算器を７連つなげたコーディングでは、制約条件の並び替えだけで最大ノード数を約半分に抑えることができた。

８、おわりに

ＣＣＶＯが択一制約条件という強い制約を優先することによって、小さな枝を生成し、それを縮めていくのに対し、本アルゴリズムは、それ自身はあまり強くない制約であるが、全体的に見ると強い制約を優先し枝を生成していくアルゴリズムであるといえる。変数の立場から見ると、頻度の高い変数を優先することによって早い段階にその変数の枝を打ち止めにすることを目的としたアルゴリズムであると思われる。

参考文献

［１］奥乃　博：二分決定グラフによる探索型組合せ問題の解法での組合せ的爆発抑制法、情報処理学会論文誌、Vol.35,No.5, pp.739-753(1994).

	変数順序				
	盤優先	逆盤優先	数優先	逆数優先	盤逆優先
最終ノード（システム全体）	81 (225)	81 (224)	81 (225)	81 (252)	81 (252)
制約順序	最大ノード数	最大ノード数	最大ノード数	最大ノード数	最大ノード数
ＣＣＶＯ	5010	4995	319650	318450	3437
即実行型	5333	5296	35758	22634	3871

図１，ＣＣＶＯによる最大ノード数および最終ノード数への影響

	変数順序				
	盤優先	逆盤優先	数優先	逆数優先	盤逆優先
最終ノード（システム全体）	81 (225)	81 (224)	81 (225)	81 (252)	81 (252)
制約順序	最大ノード数	最大ノード数	最大ノード数	最大ノード数	最大ノード数
包含による	3991	3983	23704	5105	2400

図２，最大ノード数および最終ノード数への影響