2H4-OS-03a-2

BDD構築技術を応用した組合せ最適化の一手法

Toward a BDD-based Efficient Algorithm for Combinatorial Optimization

岩下 洋哲
Hiroaki Iwashita

株式会社富士通研究所
Fujitsu Laboratories Ltd.

We discuss combinatorial optimization techniques inspired by binary decision diagrams (BDDs). There is close connection between BDDs and combinatorial optimization problems. BDDs are useful data structure to represent a large discrete space of a feasible region, where some optimization problems are formulated as shortest-path or longest-path problems. Additionally, top-down BDD construction can be viewed as binary search with memorization. It can be modified to an interesting combinatorial optimization algorithm by incorporating a branch-and-bound method into it. In this report, we review BDD-based techniques in the context of combinatorial optimization.

1. はじめに

二分決定グラフ（BDD: Binary Decision Diagram）は，離散値の集合を効率良く表現するデータ構造である [Bryant 86, Knuth 11]．一方，組合せ最適化問題は，与えられた制約を満たす離散値（実行可能解）の中で，与えられた目的関数の値を最小化（または最大化）するものを見つけ出す問題である．BDD と組合せ最適化問題には深い関係がある．BDD は問題の実行可能領域を表現することができる．その場合，ある種の組合せ最適化問題は BDD 上の最短経路（または最長経路）を求める問題に帰着できる．また，BDD は組合せ最適化問題における探索木に対応すると解釈することもできる．BDD の既約化は冗長な探索を排除する仕組みであり，動的計画法との関連性が高い [Hooker 13]．分枝限定法も組合せ最適化問題を解く代表的な手法の一つであるが，その鍵となる上界または下界の計算においても，BDD を用いたアルゴリズムが有用であることが報告されている [Bergman 13]．

組合せ最適化問題の多くは NP 困難のクラスに属するが，問題の性質をうまく利用した実用的な解法が考えられている問題が少なくない．例えば最短経路問題，巡回セールスマン問題，最大フロー問題などである．その一方，より抽象的な枠組みの中で問題を定式化し，それによって汎用性の高い最適化ツールを提供する取り組みも成功してる．線形式や二次式などで定式化された問題を解く整数計画法などが，その代表的な例である．我々は本研究において，後者に近い考え方の中で BDD を応用する良い方法を探ろうとしている．本文では，BDD の技術を組合せ最適化の観点から再検討し，性能と汎用性の高い最適化手法を実現するための考察を行う．

2. BDD と組合せ最適化

ここでは組合せ最適化問題を，離散値の定義域 $D = D_1 \times \cdots \times D_n$ に対して目的関数 $f : D \to \mathbb{R}$ および m 個の制約条件 $C_j : D \to \{0,1\}, j = 1,\ldots,m$ が与えられたとき $\min\{f(\mathbf{x}) \mid C_j(\mathbf{x}) = 1, j = 1,\ldots,m\}$ または $\max\{f(\mathbf{x}) \mid C_j(\mathbf{x}) = 1, j = 1,\ldots,m\}$ を求める問題と定義する．全ての j $(= 1,\ldots,m)$ について $C_j(\mathbf{x}) = 1$ となるような $\mathbf{x}$ の値の集合 $C \subseteq D$ は，この問題の実行可能領域と呼ばれる．単純化のため，以降では全ての i $(= 1,\ldots,n)$ について $D_i = \{0,1\}$ である問題について議論する．多値から 2 値への変換，あるいは多分決定グラフ（MDD）を考えると，ここでの議論を一般的な問題に拡張することができる．

連絡先: 岩下 洋哲，iwashita.hiroak@jp.fujitsu.com

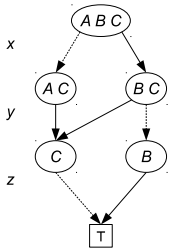

図 1: $A \wedge B \wedge C$ を表現する BDD の構築

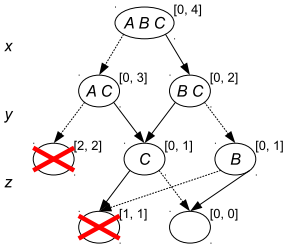

図 2: BDD 構築に分枝限定法を取り入れた最適化

まず最初に，制約条件が乗法標準形（CNF）で与えられている例を考える．論理変数 x, y, z に関して 3 つの制約条件（節）$A = x \vee y$，$B = \neg x \vee y \vee z$，および $C = \neg y \vee \neg z$ が与えられているとする．CNF から BDD への変換は伝統的な apply 演算 [Bryant 86] でも実現できるが，ここでは後に紹介する手法との関連から，節集合を「状態」として BDD を構築していく方法を取り上げる [Huang 04, 岩下 14]．図 1 は，この方法で BDD を構築した様子を描いたものである．最上位（レベル 1）にある根ノードの状態 $\{A,B,C\}$ は，どの節の充足についても確定していないことを示している．変数 x の値を 0 に定めると節 B の充足が確定し，レベル 2 の状態 $\{A,C\}$ に遷移する．反対に 1 に定めると節 A の充足が確定して，レベル 2 のもう一つの状態 $\{B,C\}$ に遷移する．レベル 2 の状態 $\{A,C\}$ で変数 y の値を 0 に定めると節 A を充足できないことが確定するため，その探索枝は打ち切って良い．このように，状態の変化をトップダウンに追跡していくことで，BDD を一気に構築できる．

目的関数が $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i)$ の形を持つとき，この目的関数は分離可能（separable）であると言う．実行可能領域を示す BDD を一度構築してしまえば，分離可能な目的関数に対しての最適化は容易である．BDD において，$x_i = 0$ に対応する枝の重みを $f_i(0)$，$x_i = 1$ に対応する枝の重みを $f_i(1)$ として，根から⊤までの最短経路または最長経路を求めれば良い．さらに [Hooker 13] では，目的関数が分離可能でない場合でも，その値が BDD における経路上の枝の重みの和で得られるような

「重み付き BDD」を構成可能であることが示されている．ただし，その効率的な（BDD を木に展開してから再び圧縮する等ではない）構成方法は自明ではなく，問題ごとにうまく設計してやる必要がある．

3. 性能の改善と汎用性の改善

BDD は実行可能領域全体を列挙するには強力な技術であるが，最適値を求めることだけを目的とする場合には，必ずしもそこまでの処理が必要ではない．最適値を含む領域，例えば目的関数の値が閾値よりも良い領域だけについて処理することが，性能改善につながるだろう．これは一種の分枝限定法と考えることができる．

前節の例において，節 A の重みを 2，節 B と C の重みを 1 とした重み付き最大充足割当問題（weighted MaxSAT）を考えてみよう．ここでは，充足されない節の重みの和を目的関数とし，その最小化を目指すものとする．BDD 構築に分枝限定法を取り入れた解法を図 2 に示す．状態遷移を計算しながらトップダウンで BDD を構築していくところは図 1 と同様であるが，図 2 の各ノードには状態の他に目的関数値の下界と上界が付与されている．どの変数値も決まっていない根ノードにおける下界は 0（全ての節が充足），上界は 4（全ての節が非充足）である．変数 x の値を 0 に定めて節 B の充足が確定すると上界が 1 減少して 3 に，1 に定めて節 A の充足が確定すると上界が 2 減少して 2 に変化する．レベル 2 の状態 $\{A,C\}$ で変数 y の値を 0 に定めると節 A の非充足と節 C の充足が確定するため，下界が 2 増加，上界が 1 減少してどちらも 2 になる．遷移先は未確定の節が無いことを示す状態 $\{\}$ である（MaxSAT 問題なので，特定の節が充足できない場合でも探索は継続する）．一般の分枝限定法と同様に，他の探索枝の上界よりも大きな下界を持つ探索枝は，その先を探索せず捨ててしまって構わない．したがってレベル 3 の他のノードが計算された時点で，このノードは不要であることがわかる．以後同様に処理していくと，レベル 4 の終端ノードまでたどり着く．終端ノードでは全ての変数値が確定しているため，下界と上界はともに目的関数の値を示す．根ノードから最小値を持つノードまでの全ての経路が解である．

上に示した方法は，MaxSAT 問題の解法としては単純すぎるのだが，より一般的な問題に対しても適用可能なものである．目的関数が定数 s_1 と中間変数 $s_2,\ldots,s_n$ を使って $f(\mathbf{x}) = \tilde{f}(s_{n+1})$，$s_{i+1} = \tilde{f}_i(s_i, x_i)$ $(i = 1,\ldots,n)$ の形で定義できるとき，アルゴリズム 1 を使用して問題を解くことができる．GETCHILD(u,d) は，ノード u の d-枝側の子ノード u_d を得る関数である．u が持つ状態を s_u とすると，u_d の状態は $\tilde{f}_i(s_u,d)$ で求められる．同じ状態を持つノードは複数の親ノードで共有される．GETBOUNDS(LB_u,u,d) は，u が持つ下界 LB_u を参照して u_d が持つ下界と上界を求める関数である．LB_u を参照可能にすることで，s_u の情報量を削減してノードの共有を起こりやすくさせられる狙いがある．

この方法では，良い下界と上界を求めることが鍵となる．MaxSAT 問題のように目的関数が m 個の部分関数の和 $f(\mathbf{x}) = \sum_{j=1}^{m} f_j(\mathbf{x})$ で定義され，それぞれの $f_j(\mathbf{x})$ が $\{x_1,\ldots,x_n\}$ の空でない（小さな）部分集合に依存しているものであれば，比較的容易に下界と上界を得ることができる．他の問題でも，BDD の緩和によって下界を見積もる [Bergman 13] の考え方が応用できるだろう．また，深さ優先探索を組み合わせれば，通常の分枝限定法のように，より良い上界を早期に発見できる可能性がある．

Algorithm 1: BDD 構築法をベースにした組合せ最適化

```
1  create root node r at level 1
2  initialize bounds [LB_r, UB]
3  for i = 1 to n do
4      foreach node u at level i do
5          if LB_u ≤ UB then
6              foreach d ∈ {0,1} do
7                  v ← GETCHILD(u,d)
8                  [lb,ub] ← GETBOUNDS(LB_u,u,d)
9                  if LB_v is not defined then
10                     add a d-edge from u to v
11                     LB_v ← lb
12                 else if lb = LB_v then
13                     add a d-edge from u to v
14                 else if lb < LB_v then
15                     remove all edges to v
16                     add a d-edge from u to v
17                     LB_v ← lb
18                 UB ← min(UB,ub)
19 return path(s) from the root to the best terminal.
```

なお，本節では制約条件（実行可能領域）の取り扱いについて詳しく触れなかったが，[Iwashita 13] で紹介されているサブセッティング法などを用いて実現することが可能である．

4. おわりに

トップダウン BDD 構築の技術を応用して組合せ最適化問題を解く方法について議論した．組合せ最適化において性能と汎用性の両立は極めて困難であるが，ここでは目的関数のクラスを限定しないアルゴリズムについて検討した．試験的な実装では実際にいくつかの興味深い問題が解けることを確認しているが，性能面ではまだ多くの課題と改善の可能性が残されているようだ．

参考文献

[Bergman 13] Bergman, D., Cire, A. A., Hoeve, W.-J. v., and Hooker, J. N.: Optimization bounds from binary decision diagrams, *INFORMS Journal on Computing*, Vol. 26, No. 2, pp. 253–268 (2013)

[Bryant 86] Bryant, R. E.: Graph-Based Algorithms for Boolean Function Manipulation, *IEEE Transactions on Computers*, Vol. 100, No. 8, pp. 677–691 (1986)

[Hooker 13] Hooker, J. N.: Decision diagrams and dynamic programming, in *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pp. 94–110, Springer (2013)

[Huang 04] Huang, J. and Darwiche, A.: Using DPLL for efficient OBDD construction, in *In Seventh International Conference on Theory and Applications of Satisfiability Testing, SAT 2004, Revised Selected Papers*, pp. 157–172 (2004)

[Iwashita 13] Iwashita, H. and Minato, S.: Efficient Top-Down ZDD Construction Techniques Using Recursive Specifications, Technical Report TCS-TR-A-13-69, Division of Computer Science, Graduate School of Information Science and Technology, Hokkaido University (2013)

[Knuth 11] Knuth, D. E.: *The Art of Computer Programming, Volume 4A, Combinatorial Algorithms, Part 1*, Addison-Wesley Professional, 1st edition (2011)

[岩下 14] 岩下 洋哲, 戸田 貴久, 津田 宏治, 湊 真一：乗法標準形で与えられた論理関数に対する二分決定グラフ構築の効率化, 人工知能学会全国大会 (2014)