ISSN 2186-5647

SAT を用いた TCAM ルールの細粒度分割及び配置

日大生産工(学部)　○好川 雄太郎
日大生産工　新井 雅之

1 まえがき

近年，サーバ仮想化技術の進歩などにより，動的なネットワーク構築，運用の需要が高まっており，ネットワークを仮想化する概念であるSDN (Software Defined Networking) が注目されている [1]. SDNを実現するOpenFlowでは，管理するスイッチのルールテーブルに大量のルールをTCAM (Ternary Content Addressable Memory) 上に配置する．TCAMではルールの並列比較を行う故に高速な処理を可能とするが，消費電力，テーブルの更新時間の問題により，配置可能なルールの数に上限が存在する[2]. この問題の対策として，ルールテーブルを複数のサブテーブルに分割し，ネットワーク上のSDNスイッチに分散して割り当てる手法が研究されている．

文献[3]では，サブテーブル割当問題を解く手法としてヒューリスティックなアルゴリズムが提案されている．また，文献[4]では，サブテーブル割当問題を命題論理式として定式化し，SAT (SATisfiability problem：充足可能性問題) として解くことによって，最適なルールテーブル分割を行う手法が提案されている．

本研究では，サブテーブルを従来手法より更に細かく分割し，各スイッチが複数のサブテーブルを保持する，ルールテーブルの細粒度分割及び配置について提案する．先行研究[4]と同様に，本問題を命題論理式として定式化し，SAT問題として解くことにより最適なサブテーブル分割数を得ることが可能である．細粒度分割によりスイッチが持つサブテーブルのサイズが小さくなり，消費電力を抑えることが可能になると期待される．

提案手法を，MiniSat [5,6]を用いて実装し，シミュレーションにより従来手法と比較し，評価を行う．

2 サブテーブル割当問題と先行研究

2.1. SDN

SDNとは，ソフトウェアによりネットワークを仮想化し，制御する概念である．SDNはSDNコントローラおよびSDNスイッチから構成され，SDNスイッチのデータの通信制御機能をSDNコントローラが一括して担う．SDNを実現しているOpenflowのネットワーク内では，SDNコントローラのみを制御することによってSDNスイッチの一元的な管理が可能となる．

Openflowで制御されるスイッチでは，パケットのヘッダ情報に対応する“フローパターン”と，出力ポートの指定や，通過またはドロップなどのフローに対する処理定義を格納している“アクション”をルールテーブルとして記憶しており，各パケットとフローパターンのマッチングにより，対応するアクションが実行される．

例として，SDNで実現されるネットワークの入り口のSDNスイッチにはACL (Access Control List) が配置されている．ACLは通信パケットを制御するためのリストで，あるパケットの通過または破棄を 1 個のルールとしてテーブルを構成する．

2.2. サブテーブル割当問題

文献[3,4]では，サブテーブル割当問題を，ネットワーク上に存在する各パスが，各スイッチに割り当てられたサブテーブルを全て経由するという条件のもとで，ルールテーブルの分割数が最大となるようにサブテーブルを配置する問題と定義する．各パスが全てのサブテーブルを経由することで，テーブルを分割しない場合と同様の制御が可能である．また，テーブルの分割数を大きくすることで，各ノードに配置するサブテーブルのサイズを小さくすることが出来る．図 1 にACLにおけるサブテーブル割当問題の例を示す．図 1 (a) は分割する前のACLテーブルの配置例を示している．図 1 (b) はルールテーブルの分割を行った際のネットワーク上でのサブテーブル配置例を示している．図の例ではACLテーブルを 3 個のサブテーブルに分割し，それらを対応するパス上の各スイッチに分散させる構成を取ることで，各スイッチのデータ量が元のテーブルの 1/3 となる．

2.3. 先行研究

文献[3]では，ルール分配におけるサブテーブル割当問題を解く手法として，ヒューリスティックなアルゴリズムであるSA (Sub-table Allocation) アルゴリズムおよびSSP (Size-balancing Sub-table Partition) アルゴリズムが提案されている．SAアルゴリズムでは，サブテーブル割当問題を彩色問

A Fine-Grained SDN Rule Table Partitioning and Distribution

Yutaro YOSHIKAWA and Masayuki ARAI

題として扱い，各パスに異なるn色が存在するようにスイッチを彩色する．SAアルゴリズムによって最大彩色数nを決定した後，SSPアルゴリズムが，最大及び最小のサブテーブルのサイズの差ができるだけ小さくなるようにしつつ，ルールテーブルをn個のサブテーブルに分割する．分割が完了した後，パス上の各スイッチにサブテーブルを割り当てる．SAアルゴリズムはヒューリスティックな手法であるため，得られる分割数が最適であることは保証されない．

Ogasawaraらは，SAアルゴリズムで用いられた彩色問題に着目し，SATソルバを用いたルールテーブルの分割を最適化する手法を提案した[4]．SATとは，ある命題論理式$P(x_1, x_2, \cdots, x_n)$に対して，$P(x_1, x_2, \cdots, x_n)$ = Trueとなる変数$x_1, x_2, \cdots, x_n$の組み合わせが存在するかという問題である．CNF (Conjunctive Normal Form) として表現された命題論理式をSATソルバに入力することで，実用規模の問題に対して高速に解を得ることができる．与えられた論理式が解を得られた場合，SATソルバは“SAT”を返し，解を得られなかった場合には“UNSAT”を返す．

先行研究[4]では，サブテーブル割当問題を2個の制約の積として表現している．1個目は，各スイッチは，彩色可能な色のうち1色でのみ彩色されるとする“one-hot制約”である．2個目は，割り当て可能な色において，各パス上のスイッチのうち少なくとも 1 個にその色が割り当てられていると判定する“出力制約”である．これら2個の制約を組み合わせてCNFを生成する．実験結果より，文献[3]で示されたヒューリスティックなアルゴリズムと比較して分割数が増大することが示されている．

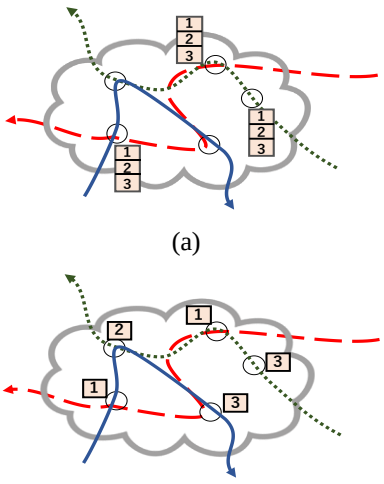

図 1. ACL におけるサブテーブル割当問題

3　ルールテーブルの細粒度分割

本研究では，従来手法を拡張し，SAT ソルバを用いたルールテーブルの細粒度分割を行う手法について提案する．

従来手法[3,4]ではルールテーブルを n 個のサブテーブルに分解し，フローパス上の各スイッチに配置している．これに対し本稿では，分割数 n を最短パス長より大きい値とするとともにパラメータ k を導入し，各スイッチに k 個のサブテーブルを割り当てる問題に拡張する．従って，1 スイッチに割当てられるテーブルのサイズは全体の k/n となる．この問題を CNF として定式化し，SAT ソルバを用いて最適解を求める．

本研究では，先行研究と同様のネットワーク構成を対象とする．ネットワーク上に存在するスイッチ数を b と置き，各スイッチを $s_1,s_2,\cdots,s_b$ と表す．また，パス数を d と置き，各パスを $p_1,p_2,\cdots,p_d$ と表す．ここで，任意のパス p_i 上に存在するスイッチの集合を $S(p_i)$と表す．分割されたサブテーブルの数を n と置き，各サブテーブルを $c_1,c_2, \cdots,c_n$ と表す．

従来手法と同様に独立変数 x_{ij} $(1 \le i \le b, 1 \le j \le n)$を用意する．この変数では，どのスイッチにどのサブテーブルが割り当てられているかを判定する．$x_{ij} = 1$ の時，スイッチ s_i にサブテーブル c_i が割り当てられていることを意味している．反対に割り当てられていない場合は $x_{ij}=1$ と表現される．

本問題では，at-most-k 制約及び出力制約を扱う．at-most-k 制約は，スイッチ s_i $(1 \le i \le b)$に対して，x_{ij} $(1 \le j \le n)$のうち高々 k 個が 1 で他が 0 となるという制約である．しかし，at-most-k 制約を SAT 問題として取り扱う場合，計算複雑度が著しく増大する[7]．そこで，本研究では，at-most-k 制約の代わりに k-hot 制約を用いる．k-hot 制約とはスイッチ s_i に対する x_{ij} のうち k 個のみ 1 とする制約である．k-hot 制約は，スイッチに割当てるサブテーブルの最大サイズにおいては at-most-k 制約と同等であり，at-most-k と比較して計算量が小さい．しかし，k-hot 制約を満たす命題論理式を等価変換して CNF を作成する際，連言節の数が指数関数的に増大してしまう．そこで，Tseitin 変換を適用して連言節の数を抑制した CNF に変換する．Tseitin 変換とは，与えられた命題論理式を，充足同値 (equi-satisifiable)な他の命題論理式に変換する手法である[8]．論理的に同値な論理式に変換する場合と比較して新たな変数の導入が必要となるものの，小さなサイズの論理式に変換できる．

以下に，4 個のサブテーブルのうち 2 個を割り当てる 2-hot 問題を取り扱う際の変換例を示す．任意のスイッチ s_i に対し，$x_{i,1}$，$x_{i,2}$，$x_{i,3}$，$x_{i,4}$ のうち 2 個が 1 であることを示す判定式 h_i を従来の

one-hot 制約の形に合わせると，

$$\begin{aligned} h_i = \{&(x_{i,1} \wedge x_{i,2}) \vee (x_{i,1} \wedge x_{i,3}) \vee \cdots \\ &\vee (x_{i,3} \wedge x_{i,4})\} \\ &\wedge \{\neg(x_{i,1} \wedge x_{i,2}) \\ &\vee \neg(x_{i,1} \wedge x_{i,3})\} \wedge \cdots \\ &\wedge \{\neg(x_{i,2} \wedge x_{i,4}) \\ &\vee \neg(x_{i,3} \wedge x_{i,4})\} \end{aligned}$$

と表現できる. CNF では連言節の内部に論理積を持つことはできないため，この形では CNF とはならない．そこで，対象の連言節またはその一部の論理積で形成されている箇所を新たな論理変数に置き換える．また，論理変数と置き換えた箇所が必要十分な関係を満たすような論理式を新たに追加する. Tseitin 変換は以上の 2 ステップで完了する. Tseitin 変換により新たに用意した変数の数は ${}_4C_2 = 6$であり，それぞれ $z_1, z_2, \cdots, z_6$ と置いたとき，変換後の判定式 h_i は，

$$\begin{aligned} h_i = (&z_1 \vee z_2 \vee \cdots \vee z_6) \wedge (\neg z_1 \vee \neg z_2) \\ &\wedge (\neg z_1 \vee \neg z_3) \wedge \cdots \wedge (\neg z_3 \vee \neg z_4) \\ &\wedge (\neg z_1 \vee x_{i,1}) \wedge (\neg z_1 \vee x_{i,2}) \\ &\wedge (z_1 \vee \neg x_{i,1} \vee \neg x_{i,2}) \wedge \cdots \\ &\wedge (\neg z_6 \vee x_{i,3}) \wedge (\neg z_6 \vee x_{i,4}) \\ &\wedge (z_6 \vee \neg x_{i,3} \vee \neg x_{i,4}) \end{aligned}$$

と置き換えることができる．上記の式は CNF を満たしているため, SAT ソルバを使って解くことが可能である．

本稿では, Tseitin 変換によって新たに用意された変数を $t_1, t_2, \cdots$ と表す．Tseitin 変換によって新たに導入される変数の数は ${}_nC_k$ となる．ここで，${}_nC_k = T$と置き，one-hot 制約を拡張させた k-hot 制約の判定式 h_i を以下に示す．

$$\begin{aligned} h_i = &\left(\bigvee_{j=1}^{T} t_j\right) \cdot \left(\bigwedge_{m=1}^{T-1} \bigwedge_{l=m+1}^{T} (\neg t_m \vee \neg t_l)\right) \\ &\wedge (\neg t_1 \vee x_{i,1}) \wedge (\neg t_1 \vee x_{i,2}) \\ &\wedge (t_1 \vee \neg x_{i,1} \vee \neg x_{i,2}) \wedge \cdots \\ &\wedge (\neg t_T \vee x_{i,T-1}) \\ &\wedge (\neg t_T \vee x_{i,T}) \\ &\wedge (t_T \vee \neg x_{i,T-1} \vee \neg x_{i,T}). \end{aligned} \quad (1)$$

出力制約は, 任意のパス $p_a (1 \le a \le d)$に対して，任意のサブテーブル $c_i (1 \le j \le n)$がパス上のスイッチ $S(p_a)$ のいずれかに割当てられなければならないという制約である．判定式を q_a として以下に示す．

$$q_a = \bigwedge_{j=1}^{n} \bigvee_{s_i}^{S(p_a)} x_{ij} \ . \quad (2)$$

式(1)，(2)より，以下の命題論理式 P を得ることができる．

$$P = \left(\bigwedge_{i=1}^{b} h_i\right) \cdot \left(\bigwedge_{a=1}^{m} q_a\right). \quad (3)$$

パケットが通過する各パス上の各スイッチには k 個のサブテーブルが割り当てられるため，$n = k$ を初期値として探索を開始する．命題論理式 P が “SAT” となった場合，n を 1 増やし，P が “UNSAT” となるまで探索を繰り返す. 限られた計算時間では, SAT ソルバが “SAT”，“UNSAT” のいずれも判断できない場合がある. 計算時間が一定の時間に達した場合には探索を打ち切り，“UNSAT” を返すように設定した．

4 評価

提案手法をMiniSat [5]を用いて実装し，シミュレーションによる評価を行った．スイッチ数bを 20 に固定し，パス数dを 1 から 20 の間で変化させた．各パス$p_a (1 \le a \le d)$ において，パス長$|S(p_a)|$を 6 に固定し，スイッチをランダムに選択した．あるパス数dにおける実験を 100 回繰り返し，分割数nの平均値を算出した．

図 2 にパス数 15 における 100 の各試行において, 提案手法および従来手法で得られた分割数を示す．図 2 より，全ての試行において従来手法(k = 1)で得られた分割数$n_{k=1}$は，4 が 73 回，5 が 27 回であったが，提案手法(k = 2)で得られた分割数$n_{k=2}$は，8 が 2 回，9 が 95 回，10 が 3 回であった．各スイッチに割り当てるサブテーブルのサイズ$(1/n_{k=1}, 2/n_{k=2})$を比較すると，100 回のうち 72 回の試行で$k/n_{k=2}$の方が小さくなることがわかった．

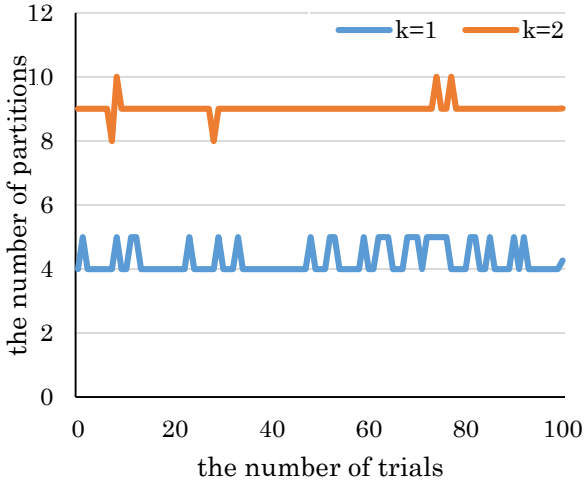

図 2. 試行回数における分割数

図 3 にパス数を 10，15，20 とした場合における 1 スイッチに割り当てられるサブテーブルの分割率(k / n)を示す．分割率は，1 スイッチに割り当てられたサブテーブルのサイズの逆数である．図 3 より，パス数 10，20 の試行では従来手法と比べ大きな変化は見られなかったが，パス数 15 における試行では，従来手法と比較して 5.5%の分割率の向上を確認できた．従って，各スイッチが持つストレージのサイズを小さくしつつサブテーブルを割り当てられる場合が存在することがわかった．従来手法ではサブテーブルのサイズが 1/4，1/5，1/6，と変化していくが，提案手法では 2/8，2/9，2/10 といった値を扱えるので，より粒度の高い分割が可能であり，各スイッチに割り当てられたサブテーブルのサイズを小さくすることが可能である．

図 4 に各パス数での平均実行時間のグラフを示す．実行結果より，パス数が増えるほどより多くの解析時間を要することがわかった．全体として，パス数の増加に伴い必要な解析時間が増加している．特にパス数d = 20 付近では，従来手法と比べ 4 秒以上の差が確認された．これは細粒度分割を行う上で扱う命題論理式が極端に複雑化したことが原因として考えられる．また，計算時間による解析の打ち切りを設定していない場合，1 回の解析で 4 時間以上掛かるケースも多く確認している．これに対処する手法として，従来手法の解析時間が短いことに注目し，従来手法と提案手法を組み合わせた手法が挙げられる．初めに従来手法による解析でUNSATとなった後，n, kを変化させて提案手法による解析を行うことで，解析時間の短縮が期待できる．

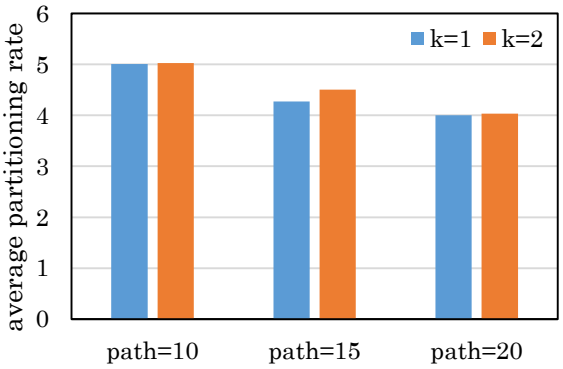

図 3. 1 スイッチが持つサブテーブルの平均サイズ

5 まとめ

本研究ではルールテーブルにおいて，SATソルバを用いた，細粒度分割による最適なサブテーブル割当手法について提案した．また本提案手法をMiniSatを用いて実験し，評価を行った．結果より，実験環境において従来手法より粒度の高い分割数を得ることができ，ストレージのサイズを小さくすることができた．今後の方針として，サブテーブル割当問題の実行時間の減少，k = 3 以上での実験および評価とその見直しを検討している．サブテーブル割当問題の定式化に着目し，CNFを構成する連言節の数を減らすことで解析時間の短縮が期待できる．また，使用するSATソルバの変更も検証する価値があると考えられる．より大きな規模のネットワークで適用できることが期待できる．

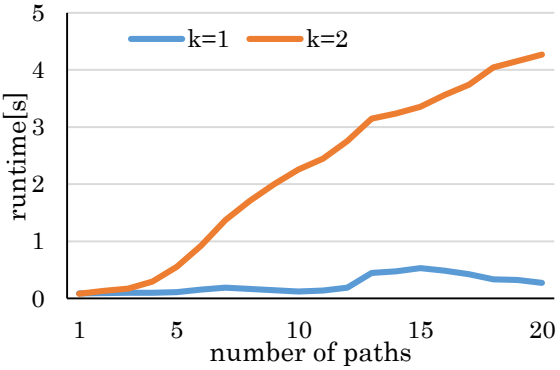

図４．各パス数での平均解析時間

参考文献

[1] N. McKeown et al., “OpenFlow: Enabling Innovation in Campus Networks,” ACM SIGCOMM Computer Communication, Vol. 38, No. 2, pp. 69–74, April 2008.

[2] H. Chen et al., “The Case for Making Tight Control Plane Latency Guarantees in SDN Switches,” ACM SOSR 2017, April, 2017.

[3] J. P. Sheu et al., “Efficient TCAM Rules Distribution Algorithms in Software-Defined Networking,” IEEE Transactions on Network and Service Management, Vol. 15, No. 2, pp. 854-865, June 2018.

[4] R. Ogasawara and M. Arai, “A SAT-Based Approach for SDN Rule Table Distribution,” PRDC 2018, FastAbstract, 2018.

[5] C. A. Tovey, “A Simplified NP-Complete Satisfiability Problem,” Discrete Applied Mathematics, Vol. 8, Issue 1, pp. 85-89, 1984.

[6] N. Eén et al., “The MiniSat Page,” [Online]. Available: http://minisat.se/

[7] A. M. Frisch and P. A. Giannoros, “SAT Encodings of the At-Most-k Constraint. Some Old, Some New, Some Fast, Some Slow,” Workshop of Constraint Modelling and Reformulation, 2010.

[8] G. S. Tseitin, “On the Complexity of Derivation in Propositional calculus,” Leningrad Seminar on Mathematical Logic, September 1966.