L-009

DAG 変形によるルール順序最適化法
A Rule Reordering Method by Modification of DAG

柴原　侑平[1) 2)]　淵野　敬[1)]　田中　賢[1)]
Yuhei Shibahara　Takasi Futhino　Ken Tanaka

1　はじめに

パケット分類とは，ルールリストとパケットを照合してパケットの振り分けを行う通信上の手続きである．専用ハードウェアを持たないネットワーク機器では，線形探索でルールとパケットの照合を行う．このためルールの増加に伴い照合回数が増加すると，通信の遅延が増大する．ポリシーを保ちつつ遅延最小のルール順序を求める問題は NP 困難であるため，様々な発見的解法が提案されている．単一機械ジョブスケジューリング問題において、根付き木となる先行制約をもつジョブでは、遅延が最も小さくなるようなシーケンスを求められることが知られている [1]．この手法をもとに，先行制約を表した DAG を根付き木にすることで遅延を減少させる．そのために先行制約グラフを遅延を減少させる根付き木にする手法を提案する．パケット分類アルゴリズムのベンチマークである ClassBench[2] を用いた計算機実験により，提案手法の有効性を確かめる．

2　パケット分類問題

表 1: ルールリストの例

Filter $\mathcal{R}$	$\lvert r_i(\mathcal{F})\rvert$
${r_1}^P = 1*1*1$	10
${r_2}^D = 1111*$	100
${r_3}^D = 1110*$	10
${r_4}^P = 1*11*$	10
${r_5}^P = 11**1$	30
${r_6}^D = *****$	20
$L(\mathcal{R},\mathcal{F})$	530

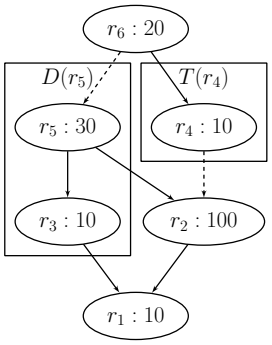

図 1: 表 1 から作成した先行制約グラフ

パケット分類問題におけるルールリストの例を表 1 に示す．パケットを長さ w のビット列とし，ルールを長さ w の $\{0,1,*\}$ で構成された文字列とする．P と D は各ルールがパケットに付与する評価型で，それぞれパケットの通過の許可と拒否を表す．到着したパケットはルールリストの１番目のルールから順番に照合し，最初に合致したルールの評価型が適用される．到着するパケットの頻度分布 $\mathcal{F}$ が与えられたとき，ルール r_i によって評価型が決まるパケットの数を，r_i の評価パケット数と呼び，$|r_i(\mathcal{F})|$ と表す．

パケットとルールとの照合を遅延 1 と考え，到着パケットの頻度分布 $\mathcal{F}$ でのルールリスト $\mathcal{R}$ の遅延 $L(\mathcal{R},\mathcal{F})$ を (1) のように定義する．

1)　神奈川大学大学院理学研究科，平塚市
Graduate School of Science, Kanagawa University, 2946, Tsuchiya, Hiratsuka-shi, Kanagawa, 259-1293, JAPAN
2)　r202170050ft@jindai.jp

定義 2.1（ルールリストの遅延）

$$L(\mathcal{R},\mathcal{F}) = \sum_{i=1}^{n-1} i|r_i(\mathcal{F})| + (n-1)|r_n(\mathcal{F})| \tag{1}$$

ルールリストのポリシーを保持するため，ルールの重複関係と従属関係を定義する．

定義 2.2（ルールの重複関係）　r_i と r_j の両方に合致するパケットが存在するとき，r_i と r_j は重複するという．

定義 2.3（ルールの従属関係）　r_i と r_j が重複しており，$i < j$ かつ評価型が異なるとき，r_j は r_i に従属するという．

ポリシーを維持するために，ルール間で守らなければいけない順序のことを先行制約という．先行制約を守りながら，遅延が最小になるルール順序を求める問題を，ルール順序最適化問題という．ルール同士の先行制約をエッジとすることで，先行制約を図 1 のような DAG で表現することができる．

3　根付き木への変形法

単一機械ジョブスケジューリング問題において、根付き木となる先行制約をもつジョブでは、遅延が最も小さくなるようなシーケンスを求められることが知られている [1]．そこで，与えられたルールリストをもとに先行制約グラフを作成し，先行制約を根付き木に変形することを考える．根付き木に変形する際に，評価パケット数の大きなルールが上位に来るように変形する。そのために，前後のルールの重みも同時に考慮することで、適切な根付き木を構成する方法を提案する。

3.1　ルールの比較による入次数 1 への変形

提案手法は先行制約を根付き木にするために，従属関係を守りながら入次数が 2 以上の全てのルールを入次数 1 に変形する手法である．ルール番号が大きいルールから入次数 2 以上のルールの探索を始め，入次数 2 以上のルールに従属している r_i と r_j の比較をおこなう．比較をおこなうために，r_i を配置することにより影響を受けるルールを集合に加え，その平均値を比較に用いることで，r_i を配置することにより影響を受けるルールも考慮に入れた上で比較をおこなうことができる．r_i から到達可能なルールの中で，r_i より評価パケット数が大きいルールは，r_i の位置に影響されず，r_i の平均値を求める際に無視できる．また，r_i に到達可能なルールの中で，r_i より評価パケット数が小さいルールは r_i の位置に影響を与えないので，r_i の平均値を求める際に無視できる．これらを踏まえて r_i と r_j の比較をおこなう．

$D(r_i)$ を，r_i から到達可能なルールの中で，評価パケット数の平均値が最小になるようなルール集合とする．ただし，r_i と r_j の共通部分はルール集合から除

Algorithm 1 Rule set with the highest average in the reachable rules

Input: r_i
Output: $T(r_i)$
1: $T(r_i), Tmp \leftarrow$ new Hash Set$\{r_i\}$
2: $r_k \leftarrow r_i$
3: $r_i^{-1} :\Leftrightarrow$ the rule that depends on r_i
4: **while** $1 \leq$ InDegree of r_k **do**
5: **if** rules reachable to r_j contains r_k **then**
6: **break**
7: **end if**
8: insert r_k^{-1} into Tmp
9: **if** $Tmp/|Tmp| > T(r_i)/|T(r_i)|$ **then**
10: $T(r_i) \leftarrow tmp$
11: **end if**
12: $r_k \leftarrow r_i^{-1}$
13: **end while**
14: **return** $T(r_i)$

く．集合 $T(r_i)$ を，r_i から，r_i と r_j の共通部分もしくは *source* ルールまでの中で，評価パケット数の平均が最大になるようなルール集合に $D(r_i)$ を加えた集合とする．$T(r_i)$ に含まれるルールの重みの和を，$T(r_i)$ に含まれるルールの数で割った値 $A(r_i)$ を，r_i の評価値とする．次に $A(r_i)$ と $A(r_j)$ の比較をおこない，評価値の低いルールから，元の入次数 2 以上のルールへの推移辺を削除し，評価値の高いルールへの有向辺を追加する．有向辺を追加することで新しく入次数 2 以上のルールができる場合は，そのルールに対して比較の動作をおこなう．この比較を入次数 2 以上のルールがなくなるまでルール番号の大きいルールから順に再帰的におこなう．根付き木となった先行制約グラフに，文献 [1] の手法を用いることで，そのグラフのなかで遅延が最小になるルールリストを求めることができる．

3.2 時間計算量

Algorithm1 では $T(r_i)$ を求めている．r_i に n 個のルールが到達可能な場合，この動作は高々 n 回おこなうので，計算量は $O(n)$ である．また，r_i が n 個のルールに到達可能な場合，$D(r_i)$ を求める計算量は $O(n^2)$ である．上記の動作を，有向辺を追加した後に入次数 2 以上となるルールが存在しなくなるまで繰り返す．この動作は高々 n 回なので、計算量は $O(n)$ となる．また，先行制約グラフの入次数 2 以上のルールに従属するルールの数だけ比較をおこなう．n 個の入次数 2 以上のルールそれぞれに，n 個のルールが従属している場合，この動作は高々 n 回なので，計算量は $O(n^2)$ となる．根付き木からルールリストを作成する手法 [1] の計算量は $O(n^2)$ である．そのため，この手法の計算量は $O(n^5)$ となる．

3.3 動作例

表 1 の場合，従属関係に基づき図 1 のような先行制約グラフが作成される．ルール番号が一番大きい r_6 から入次数が 2 以上のルールの探索を始める．r_2 の入次数が 2 なので，r_2 に従属している r_4 と r_5 の比較をおこなう．図 1 の例では，$D(r_4) = \{r_4\}$, $D(r_5) = \{r_3, r_5\}$ であり，$T(r_4) = \{r_4\}$, $T(r_5) = \{r_3, r_5\}$ となる．これにより，$A(r_4) = 10, A(r_5) = 20$ となる．$A(r_5) > A(r_4)$ なので，r_4 から r_2 への推移辺を削除し，r_4 から r_5 への有向辺を新たに追加することで，図 2 の従属グラフになる．図 2 のグラフにおいて，r_6 から r_5 の辺は推移辺なので削除する．これを入次数 2 以上のルールがなくなるまで繰り返し，図 3 になる．(1) により，表 1 のルールリストの遅延は 530 だが，表 1 に対し提案手法を適用した結果，遅延は 510 に減少する．

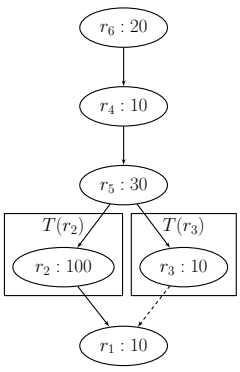

図 2: r_2 の入次数を 1 にした先行制約グラフ

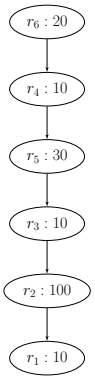

図 3: 根付き木となった先行制約グラフ

4 計算機実験

提案手法の有効性を確認するために計算機実験を行った．実験には Class Bench[2] を用いてルール数 1000 から 5000 までの acl のルールリストを各 10 個生成した．この問題に対する発見的解法の中で平均的に最も遅延を減少させる SGM(Sub-Graph merging)[3] と提案手法で並び替えをおこない，それぞれのルールリストの遅延と並び替え時間を計測した．表 2 はそれぞれの手法で並び替えた acl のルールリストの遅延の平均，表 3 は並び替えにかかった時間である．acl の全てのルール数において SGM より遅延を減少させることができた．

表 2: acl の遅延の平均

$\|\mathcal{R}\|$	SGM	提案手法
1000	2.51717e+07	2.43635e+07
2000	4.83311e+07	4.6807e+07
3000	5.85689e+07	5.83877e+07
4000	7.9911e+07	7.7788e+07
5000	9.06849e+07	9.02561e+07

表 3: acl の実行時間の平均

$\|\mathcal{R}\|$	SGM	提案手法
1000	3.06679e+08	1.53036e+09
2000	1.44765e+09	1.5694e+10
3000	3.65784e+09	5.50453e+10
4000	7.12631e+09	1.50786e+11
5000	1.55297e+10	3.50166e+11

5 まとめと今後の課題

提案手法は，ルール数 1000 から 5000 の acl において SGM よりも最大で 3% ほど遅延を減らした．今後の課題は，acl 以外のルールにおいての実験，タイトな計算量の算出，計算量の短縮，ルールリストが複雑な場合に適切に遅延を減少させることができない場合の解決が挙げられる．

参考文献

[1] W A. Horn. Single-machine job sequencing with tree-like precedence ordering and linear delay penalties. *Siam Journal on Applied Mathematics - SIAMAM*, Vol. 23, , 09 1972.
[2] David E. Taylor and Jonathan S. Turner. Classbench: A packet classification benchmark. *IEEE/ACM Trans. Netw.*, Vol. 15, No. 3, pp. 499–511, June 2007.
[3] A. Tapdiya and E.W. Fulp. Towards optimal firewall rule ordering utilizing directed acyclical graphs. In *Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th Internatonal Conference on*, pp. 1–6, Aug 2009.