アルゴリズム 68－7
（1999．5．10）

木型ネットワークにおける単一コータリーの二基準配置問題に関する数値実験

田中章裕* 木庭淳† 入江豪‡

1999年5月10日

概要

コータリーは相互排除問題を解決するための手段の1つであり、木型ネットワークでは耐故障の観点で最適なコータリーは単一頂点となることが知られている。我々は木型ネットワークにおいて、期待生存度および負荷分散の2つの評価基準に関するコータリー配置問題を考察する。まずランダムに木を発生させて期待生存度を最大にする頂点と負荷分散を最小にする頂点との平均距離を調べ、2つの基準を共にある程度満足するような頂点を見つける。次にシミュレーションによりその正当性を確認する。またコータリーの処理能力の限界についても調べる。

1 はじめに

複数プロセスが一つの資源を共有するような分散システムでは、どれか一つのプロセスだけが資源の使用を許可され、他のプロセスはその使用が終わるのを待たねばならない。このような問題を相互排除問題と言い、コータリーはこの問題を解決する手段の一つである [5]。

U をプロセスの集合とすると、以下の条件を満たすような U の部分集合 Q_i $(Q_i \subseteq U, 1 \leq i \leq N)$ をコーラムといい、コーラムの集合をコータリーという。

1. 任意の i と j $(1 \leq i,j \leq N)$ に対して、$Q_i \cap Q_j \neq \phi$ である。
2. 任意の i と j $(1 \leq i,j \leq N, i \neq j)$ に対して、$Q_i \not\subseteq Q_j$ である。

すなわち、任意の二つのコーラムには必ず共有するプロセスが存在し、あるコーラムが別のコーラムに含まれてはならない。

コータリーにより相互排除問題を解決するしくみは、概略的には以下のようである。資源を利用しようとするプロセス A は、任意のコーラム Q_i 中のすべてのプロセスから許可を得て初めて資源の使用が可能となる。プロセス A が資源使用のリクエストを発行中、もしコーラム Q_i 中にプロセス B に対して資源使用許可を与えているプロセスがあれば、プロセス A に対する許可は与えられない。上記1の性質よりプロセス A がリクエストを送っているコーラム Q_i とプロセス B がリクエストを送っているコーラム Q_j には必ず共有プロセスが存在するので、資源の二重使用を避けることができる。

コータリーを評価するために従来から様々な基準が提案されている。少なくとも一つのコーラム中のプロセスがすべて故障していない確率すなわち可用度による評価 [4, 10]、可用度と類似しているが故障していないプロセス集合が最大となるような期待生存度による評価 [9]、ネットワークの混雑を避けるための負荷分散による評価 [3]、生存コーラムを探すコストを少なくするような探索複雑度による評価 [11]、などがある。

木型ネットワークにおけるコータリーについて我々は負荷分散および期待生存度に着目する。Papadim-

*コベルコシステム
†神戸商科大学管理科学科
‡オービック・オフィスオートメーション

itriou ら [9] は期待生存度に関して最適な形状は単一コータリーであることを述べているが、負荷分散に関しても単一コータリーが最適となるのは明らかである。従って我々はまずコータリーからの距離により負荷分散と期待生存度を単純にモデル化した評価式 [7, 9] を用い、ランダムに木を発生させて期待生存度を最大にする頂点と負荷分散を最小にする頂点との平均距離を調べる。次に 2 つの評価基準を共にある程度満たすようなコータリーの配置問題について考察する。さらにその結果をシミュレーションによって評価し、コータリーの処理能力の限界を測る。

2 簡易モデル

2.1 諸定義

まず、負荷分散についての評価式を定義する。木 $T=(V,E)$ が与えられたとき $d(v,c)$ を T 上の頂点 $v,c\in V$ 間の距離すなわち辺数とし、a_v をアクセス比率とする。このとき、コータリー c の負荷分散係数 $L(c)$ を文献 [7] と同様、

$$L(c)=\sum_{v\in V} a_v\cdot d(v,c)$$

とする。本論文では、アクセス比率を 1 として考える。

次に期待生存度についての評価式を定義する。故障確率を p とすると、コータリー c の期待生存度 $W(c)$ を文献 [9] と同様、

$$W(c)=\sum_{v\in V}(1-p)^{d(v,c)}$$

とする。期待生存度とは、頂点が故障する事を考えた場合、どの位頂点が生き残れるかを示す基準である。負荷分散が最小になる頂点を L_{min} とし、期待生存度が最大になる頂点を W_{max} と表す。

ここで $r\in V$ を根とする部分木 T_r に対して頂点 $v\notin T_r$ が r に隣接しているとき、T_r を "v から発する部分木" と呼ぶことにする。木 T のセントロイド c_e とは c_e から発する最大サイズの部分木 $T_m=(V_m,E_m)$ が $|V_m|\le(|V|-1)/2$ を満たすようなものをいう。もしすべての頂点 v についてアクセス比率 $a_v=1$ の場合、Kang ら [6] により次の事実が知られている。

定理 2.1 *[6]* 与えられた木 $T=(V,E)$ において $\min_{c\in V}L(c)$ を満たすような頂点 c (すなわち L_{min}) は木 T のセントロイド c_e と一致する。 ∎

この定理をもとにして次節で負荷分散を最小にする頂点を求める。

2.2 平均距離と二基準での配置

我々の興味は、第 1 に L_{min} と W_{max} が平均的にはどのくらい離れるかということであり、第 2 に 2 つの基準を共にある程度満たすような頂点を探すことである。ここでは数値実験の方法について述べる。

procedure 平均距離の測定
  for 故障確率 p を変化させて繰り返し
    for n 回繰り返し
      ランダムな木を発生させる。
      L_{min} を決定する。
      W_{max} を決定する。
      L_{min} と W_{max} の距離を測る。
    end
    平均距離を求める。
  end
end.

手順 1 : ランダムな木を発生させる。
文献 [1] 参照。

手順 2 : 負荷分散について最適な頂点を求める。
セントロイドを求める方法 [7] によって負荷分散を最小にする頂点 L_{min} を決定する。

手順 3 : 期待生存度について最適な頂点を求める。
全ての頂点について、定義した式によって評価を行い、一番値が大きかった頂点を期待生存度について最適な頂点 W_{max} とする。

手順 4 : L_{min} と W_{max} の距離を測る。
手順 2 と手順 3 で求めた L_{min} と W_{max} の距離を測定する。

手順１～４を繰り返し、平均距離を求める。さらに故障確率を変化させる。

手順 A：χ^2 検定を行い、発生させた木が一様であるかどうかを調べる

ランダムな木の次数の理論値 [8] と、実際に発生させた木の次数の実測値との間のずれを調べる為に χ^2 検定を行う。これにより平均値を計算するために必要な回数 n を設定できる。

手順 B：αL_{min} 値の範囲内で期待生存度が最も高い頂点を求める。

負荷分散と期待生存度の統合点を求める。L_{min} 値を α 倍だけ緩め、その範囲内で期待生存度が最大となる頂点を統合点とし、２つの基準に関する配置点とする。

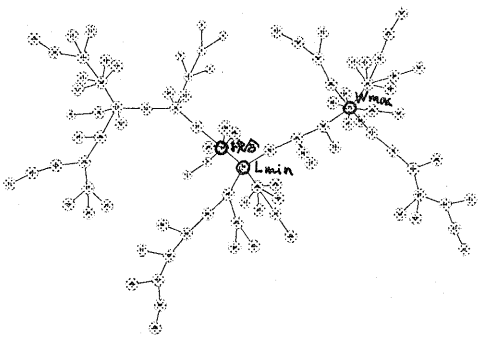

図 1: ランダムに発生した木（頂点数 100）の一例

L_{min} と W_{max} の平均距離については図 3 の結果を得た。図 1 の例での L_{min} と W_{max} および統合点の位置を図中に示す。なお χ^2 検定により $n = 200$ で充分であることが分かっている。

3 シミュレーション

3.1 シミュレーションモデル

前節の簡易モデルの実験（手順 B）で得た一つの木の頂点にコータリーを固定してシミュレーションを行う。比較のために他の頂点にもコータリーを配置した実験も行う。

また、頂点数の異なる木を用いた結果を比較する。このときのコータリーの位置はそれぞれ簡易モデルにより決定したものを用いる。こちらは、コータリーの処理能力を評価するために行なう。

シミュレーションモデルでは以下の仮定をおく。

1. リクエストの処理順は FCFS とする。
2. コータリーから要求プロセスへのリプライは行わない。
3. 各頂点間の転送時間は一定とする。
4. リクエスト、及び故障はどの頂点にも同じ確率で発生する。

上記仮定 2 について、リプライは発生頂点へリクエストと全く同じルートを通り送られるため、各頂点の使用回数は片道の頂点使用回数のちょうど２倍となる。よって比較の際の影響はないのでリプライは省略している。

シミュレーションは、故障の発生、リクエストの発生、コータリーの処理の３つの段階からなる。コータリーで一つ処理が行われるたびにリクエストを発生させ、時刻を進める。リクエスト、故障については別の時計を用意し、コータリーで進められる時刻と連動するようにしている。

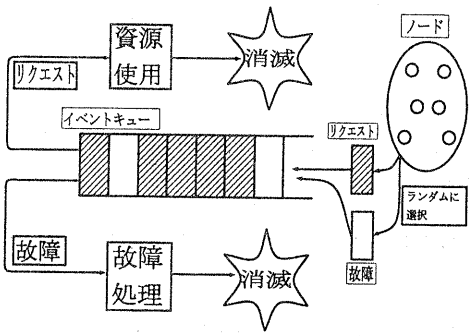

図 2: シミュレーションの流れ

リクエストは使用可能な全頂点から確率 1/2 で発生する。故障は使用可能な全頂点からランダムに選択される。発生したイベント（リクエストまたは故障）は発生頂点からコータリーの位置までの転送時

間をプラスされ、コータリー到着時刻を求めた後、コータリーのイベントキューに到着時刻順に挿入される。キューから取り出したイベントがリクエストのときには、指数分布に従う資源使用時間を決める。取り出したイベントが故障のときには、故障頂点以下、使用不可能になった頂点にチェックを入れる。以後、チェックされた頂点からはイベントは発生せずその使用はできない。

シミュレーションの終了条件は3通りある。第1は規定数（3000個）のリクエストを処理したとき、第2は故障のために使用不可能となった頂点が全頂点数 $|V|$ の半分以上に達した場合、第3はコータリーの故障時に終了する。

3.2 評価基準

平均負荷

リクエストが発生しコータリーまで転送される際には、発生頂点からコータリーのある頂点まで頂点に沿って送られる。このときの頂点の平均使用回数を測定する。ただし故障のために早期から使用されなくなる頂点もあるため、頂点の使用回数はその頂点が使用されていた時間の単位時間で計算する。

$$\text{平均負荷} = \frac{\text{単位時間の頂点使用回数の和}}{\text{全頂点数}}$$

最大負荷差

単位時間あたりの頂点使用回数の最大と最小の差である。

$$\text{最大負荷差} = \text{頂点使用回数の最大値} - \text{最小値}$$

最大生存度（期待生存度）

万一故障が起きても、どのくらいの頂点が使用可能かを測るための基準である。

$$\text{生存度} = \frac{\text{使用可能頂点数}}{\text{故障頂点数} + \text{使用可能頂点数}}$$

$$\text{最大生存度} = \frac{\text{生存度の和}}{\text{故障回数}}$$

生存度は故障の発生のたびに求め、シミュレーション終了時に最大生存度を求める。

平均待ち時間

リクエストがコータリーに届いてから、処理されるまでにかかった時間の平均である。

$$\text{平均待ち時間} = \frac{\text{リクエストの待ち時間の和}}{\text{処理リクエスト数}}$$

4 結果とまとめ

以上のような実験の結果、予想通り L_{min} と W_{max} の距離は故障確率が高くなるにつれて離れていくことが明らかになった（図3）。しかし通常は故障確率はかなり小さいと仮定してよいため、L_{min} の点は期待生存度も大きな値を持つことを意味している。またシミュレーションによる実験で、簡易モデルによって決定した位置にコータリーを置いた場合、負荷分散および期待生存度の両基準共に優れていることが確認できた。（図4、5、6）そして頂点数の違いによる処理能力の比較から、処理限界についての目安を得ることができた。

参考文献

[1] Alonso,L. and Schott,R., "Random Generation of Trees", *Kluwer Academic Publishers*, 1995.

[2] Garcia-Molina,H. and Barbara,D., "How to Assign Votes in a Distributed System", *Journal of the ACM, 32* (4): 841–860, 1985.

[3] Holzman,R., Marcus,Y. and Peleg,D., "Load balancing in quorum systems", *SIAM Journal on Discrete Mathematics, 10* (2):223–245, 1997.

[4] Ibaraki,T., Nagamochi,H. and Kameda,T., "Optimal coteries for rings and related networks", *Distributed Computing, 8* : 191–201, 1995.

[5] 亀田恒彦, 山下雅史, "分散アルゴリズム", 近代科学社,1994.

[6] Kang,A.N.C. and Ault,D.A., "Some Properties of a Centroid of a Free Tree", *Information Processing Letters, 4* (1):18–20, 1975.

[7] Kiniwa,J. and Nagamochi,H., "Optimal Location of Singleton Coterie in Tree Networks", under manuscript.

[8] Palmer,E.M., "Graphical Evolution: An Introduction to the Theory of Random Graphs", *Wiley*, 1985.

[9] Papadimitriou,C.H. and Sideri,M., "Optimal Coteries", In *Proceedings of 10th ACM Symposium on Principles of Distributed Computing*, 75–80, Aug. 1991.

[10] Peleg,D. and Wool,A., "The availability of quorum systems", *Information and Computation, 123* (2): 210–223, 1995.

[11] Peleg,D. and Wool,A., "How to be an efficient snoop, or the probe complexity of quorum systems", In *Proceedings of 15th ACM Symposium on Principles of Distributed Computing*, 290–299, 1996.

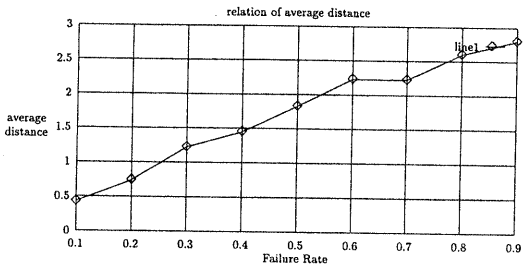

図 3: L_{min} と W_{max} の距離

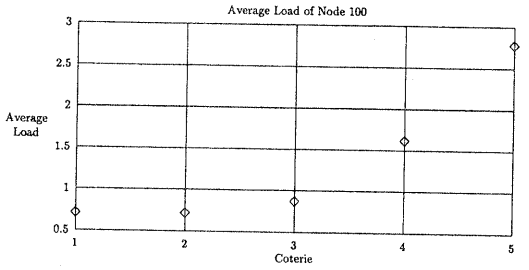

図 4: 平均負荷

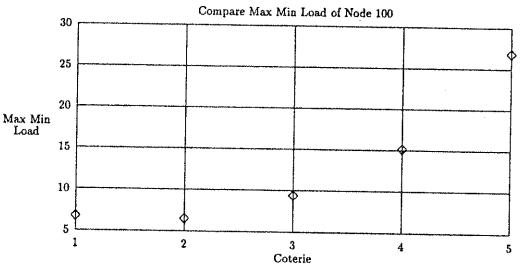

図 5: 最大負荷差

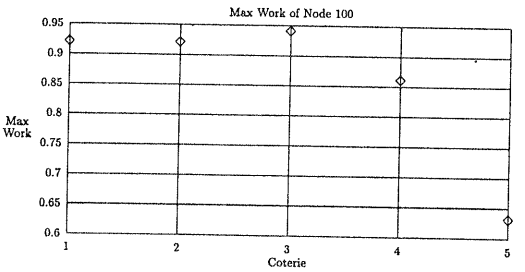

図 6: 期待生存度