# 分散システムの負荷均一化における反復解法について
# Iterative Methods for Load Balancing in Distributed Systems

林 幸雄 （北陸先端科学技術大学院大学） yhayashi@jaist.ac.jp
Yukio HAYASHI (Japan Advanced Institute of Science and Technology)

キーワード: 拡散方程式, 多項式スキーム, 行列の分離, 2 次計画問題
Keywords: Diffusion Equation, Polynomial Scheme, Splitting of Matrix, Quadratic Programming Problem

## 1 はじめに

コンピュータネットワークの性能や規模の急成長に伴い, インターネット上などに散在する複数台のマシン[1] による分散コンピューティングが近年注目を集めている [9]. その中で, 負荷均一化問題は重要な課題の 1 つである. これは, ネットワーク上の局所処理のみによって, いかにして迅速に負荷移動を行い, 均一な負荷配分にするかという問題で, これまで主に並列計算機に対して効率的な計算スキームが検討されてきた [11]. しかしながら, 分散システムは並列計算機とは異なる特徴：疎結合, 物理的論理的なプロセスの独立性, ヘテロ性（各サーバの処理能力や経路ごとにデータ転送速度が異なる）[9] などを持ち, 基本的に非同期な処理が前提となる.

本報告では, ネットワーク上に分散したサーバ間の局所処理によって負荷均一化を行う拡散法 [11] が, 多項式による反復, 連立方程式, 2 次計画問題などのさまざまな形式に帰着することを考え, 効率的な処理を行うには, 負荷の均衡のみならず無駄な巡回フローを抑えることが本質的に重要であることを指摘する. さらに, 巡回路を持たない全域木を考えれば, 従来の計算スキームのような（同期的な）反復処理をする必要がなく, メッセージ伝搬によって直接的に負荷移動量が求められるとともに, 分散システムに適した非同期な負荷移動が効率的に実現できることを示す.

[1] PC やサーバなど性能の異なるさまざまなコンピュータをさすが, 本報告では一貫してサーバと称する.

## 2 拡散方程式の反復解法

ネットワーク全体が均一になるようにサーバ間の負荷量の差を拡散伝搬していく, 拡散法における多項式スキームの概要を述べ, 行列の分離法の可能性についても触れる.

### 2.1 従来の多項式スキーム

多重辺や各頂点の自己ループがない無向グラフ $(V, E)$ に対する, 重み付き離散 Laplacian [1]

$$L = BWB^T,$$

による拡散方程式を考える. ここで, 各辺 $e \in E$ の重みの対角行列 $W \stackrel{\rm def}{=} diag(\omega_e)$, $m \times n$ 接続行列 $B$, $|V| = m$, $|E| = n$ とする.

その拡散方程式の差分版

$$\mathbf{f}^k = (I - \Delta t L)\mathbf{f}^{k-1} = F^k \mathbf{f}^0, \qquad (1)$$

$F \stackrel{\rm def}{=} I - \frac{1}{\kappa}L$, $\frac{1}{\kappa} \stackrel{\rm def}{=} \Delta t > 0$, $k$ 回目の値（各頂点 $u \in V$ におけるサーバの負荷量に相当）を表す $\mathbf{f}^k = (f^k(1), \ldots, f^k(m))$, に対して, 多項式を用いた以下の反復解法が知られている [4].

$p_0(t) = 1$, $p_1(t) = t$, $0 < \beta < 2$,

$$p_k(t) = \beta\, t\, p_{k-1}(t) + (1 - \beta)\, p_{k-2}(t).$$

行列 $F$ に関する多項式 $p_k(F)$ を考えると,

$$\mathbf{f}^1 = p_1(F)\mathbf{f}^0 = F\mathbf{f}^0,$$

$$\mathbf{f}^k = p_k(F)\mathbf{f}^0 = \beta F \mathbf{f}^{k-1} + (1 - \beta)\mathbf{f}^{k-2}. \qquad (2)$$

ここで, $L$ の $l$ 個の相異なる固有値 $0=\lambda_1<\lambda_2<\ldots<\lambda_l$ に対して, $\lambda'_i \stackrel{\rm def}{=} \frac{\lambda_i}{\kappa}$ とし[2], $F$ の固有値 $-1<\mu_l<\ldots<\mu_i<\ldots<\mu_1=1$,

$$\gamma \stackrel{\rm def}{=} \max_{i>1}|\mu_i| = \max_{i>1}|1-\lambda'_i| < 1,$$

から, (2) は 1 ステップ前の値のみを用いる FOS: $\beta=1$, 最適加速パラメータ $\kappa_{opt}=\frac{\lambda_2+\lambda_l}{2}$ [2] や[3], 2 ステップ前の値も用いる SOS: $\beta_{opt}=\frac{2}{1+\sqrt{1-\gamma^2}}$, Chebyshev スキーム:

$$\beta_1=1,\ \ \beta_2=\frac{2}{2-\gamma^2},\ \ \beta_k=\frac{4}{4-\gamma^2\beta_{k-1}},$$

を含む. Chebyshev 多項式の漸化式

$$C_0(x)=1,\quad C_1(x)=x,$$

$$C_{k+1}(x)=2xC_k(x)-C_{k-1}(x),\ \ (k\geq 1)$$

を用いれば,

$$\beta_k=\frac{2C_k(\frac{1}{\gamma})}{\gamma C_{k+1}(\frac{1}{\gamma})},$$

である. このとき, (2) は Chebyshev 準反復法 [10] に他ならない. 通常, 収束速度は FOS, SOS の順である（Chebyshev スキームは漸近的 $k\to\infty$ には SOS と同等 [4]）.

さらに一般化された最適な多項式を用いる OPS: $p_0(t)=1$, $p_1(t)=\frac{1}{\gamma_1}[(\alpha_1-t)p_0(t)]$,

$$p_k(t)=\frac{1}{\gamma_k}\left[(\alpha_k-t)p_{k-1}(t)-\beta_k p_{k-2}(t)\right], \quad (3)$$

$$\alpha_k=\frac{<t\,p_{k-1},\,p_{k-1}>}{<p_{k-1},\,p_{k-1}>},\quad (k=1,\ldots,l-1),$$

$$\beta_k=\gamma_{k-1}\frac{<p_{k-1},\,p_{k-1}>}{<p_{k-2},\,p_{k-2}>},\quad (k=2,\ldots,l-1),$$

$\gamma_1=\alpha_1-1$, $\gamma_k=\alpha_k-1-\beta_k$, $(k=2,\ldots,l-1)$ では, $F$ の固有値による $\nu_j=1-\mu_j$, $(j=2,\ldots,l-1)$ を選べば $l-1$ 回の反復で収束することが保証されている [4]. ここで, 各 $k$ 次の多項式は $p_k(1)=1$ をみたす直交系 $<p_i,p_k>=0$, $(i\neq k)$ で, 内積は

$$<p,q>\stackrel{\rm def}{=}\sum_{j=2}^{l}\nu_j p(\mu_j)q(\mu_j),$$

である（$\nu_j>0$）. 反復式は (3) の $p_k(F)$ より,

$$\mathbf{f}^1=\frac{1}{\gamma_1}\left[\alpha_1\mathbf{f}^0-F\mathbf{f}^0\right],$$

$$\mathbf{f}^k=\frac{1}{\gamma_k}\left[\alpha_k\mathbf{f}^{k-1}-F\mathbf{f}^{k-1}-\beta_k\mathbf{f}^{k-2}\right],$$

これらは各頂点において隣接頂点との局所処理で実行できる.

一方, $l-1$ 回の反復で収束性を保証する, OPS よりも簡単な計算スキーム OPT:

$$\mathbf{f}^k=\left(I-\frac{1}{\lambda_{k+1}}L\right)\mathbf{f}^{k-1},$$

は並列計算機に適したグラフのデカルト積の構造や, 各プロセッサ（サーバ）の性能が異なるヘテロなネットワークにまで適用できる [5].

OPS や OPT は計算スキームとしてはかなり確立されたものと考えられるが, 例え負荷移動量が求まってもマイナスの負荷量は実在し得ないので各辺上の**負荷移動の順序をどうするかが問題**となる. また, 予め Laplacian $L$ の固有値を求めておく必要があり, 特にネットワーク（グラフ）構造が変化しうる環境では, **計算量的にもネック**となる. 拡散法 (1) と等価な連立方程式 (7) を直接的に解く共役勾配法も同様に有限反復であるが [4], 上記のスキームのように**ネットワーク上の局所処理では実行できない**のでさらに深刻である. いずれにしても, Laplacian の相異なる固有値の個数や値が本質的に拡散伝搬の収束速度に影響すると考えられ, それらはグラフの構造（半径やボトルネック部分など）に依存する [1].

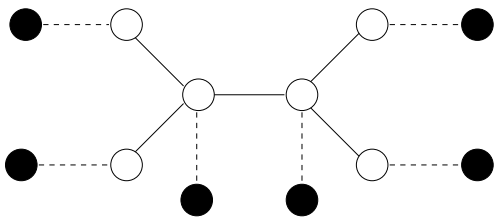

図 1: 拡張グラフ： ○ は各サーバ, ● はダミー頂点, 点線はそれらを結ぶ重み付き辺を表す.

## 2.2 行列の分離法の適用

図 1 のように, 各頂点 $u\in V$ のサーバをミラーリングするダミー頂点 $u'$ と, それらを結ぶ各辺の重

[2] 重みを $W' \stackrel{\rm def}{=} \frac{1}{\kappa}W$ とした Laplacian $BW'B^T$ を考えていることに相当する.

[3] $\min_\kappa\gamma=\min_\kappa\max_{i>1}|1-\frac{\lambda_i}{\kappa}|$ より $\kappa_{opt}$ を得る.

み $\kappa$ を持つ拡張グラフを考える. このとき, (1) の解 $L\mathbf{f}=0$ は Dirichlet 的な（各反復で変動する）境界値 $f(u')$ を持つ,

$$-\sum_{v\sim u}\omega_e(f(v)-f(u))-\kappa(f(u')-f(u))=0,$$

を解くこと, すなわち, 連立方程式

$$A\mathbf{f}=\mathbf{c},$$

$A \stackrel{\text{def}}{=} \kappa I+L$, $c(u) \stackrel{\text{def}}{=} \kappa f(u')=\kappa f(u)$ に帰着する. ここで, $e=(u,v)$, $v\sim u$ は頂点 $u$ の隣接点を表す.

行列の分離 [8] として, $A=M-N$, $M=\kappa I$, $N=-L$ を考えると,

$$\mathbf{f}=M^{-1}N\mathbf{f}+M^{-1}\mathbf{c}, \tag{4}$$

を得る. このとき, $M^{-1}N=-\frac{1}{\kappa}L$, $M^{-1}=\frac{1}{\kappa}I$ の作用はネットワーク上の局所処理で実行できる[4]. 但し, 変動境界値なので, 通常の固定境界値でのスペクトル半径に関する収束条件 $\rho(M^{-1}N)=\frac{\lambda_1}{\kappa}<1$ は必要なく, パラメータ値を $\kappa_{opt}$ とするのが妥当と考えられるが, それでもこれは FOS の変種であるため収束速度は遅い.

一般に, $\mathbf{c}=M\mathbf{f}$, $M^{-1}N=-\Delta t\times L$, を満たす行列の分離 $M$, $N$ が見つかれば, (1) は (4) に帰着する. 但し, $L$ の対称性から, $(M^{-1}N)^T=M^{-1}N$, すなわち, $M$,$N$ が対称ならば可換 $MN=NM$ でなければならない. 残念ながら, SOR 法などは該当せず, これらを満たすより高速な新たな計算スキームの発見に期待したい.

# 3 分散システム上の負荷移動

拡散法 (1) と等価な 2 次計画問題から, **無駄な巡回フローに関する問題点**を指摘し, 分散システム上の非同期で局所的な処理に適した効率的なアルゴリズムについて議論する.

## 3.1 無駄な巡回フロー

さて, 巡回路を持つグラフ上の負荷移動を考える.

[4] 式 (4) の行列分離に Jacobi 法を適用した場合も, 局所処理で実現できる.

ステップ数 $k$ に関する各辺 $e\in E$ 上の負荷移動量 (フロー）の累積を変数 $z_e$ で表せば, (1) は以下のように書き直せ,

$$y_e^{k-1}=-\omega_e(f^{k-1}(v)-f^{k-1}(u))\times\Delta t,$$

$$z_e^k=z_e^{k-1}+y_e^{k-1},\quad z_e^0=0,$$

$$f^k(u)=f^{k-1}(u)-\sum_{e,\bar{e}}y_e^{k-1},$$

これらはまた 2 次計画問題

$$\min\quad \tfrac{1}{2}\mathbf{z}^TW^{-1}\mathbf{z}, \tag{5}$$

$$s.t.\qquad B\mathbf{z}=\mathbf{b}, \tag{6}$$

と等価となる [7]. ここで, $\mathbf{b} \stackrel{\text{def}}{=} \mathbf{f}^0-\bar{\mathbf{f}}$, $\bar{\mathbf{f}}$ は均一解 $\bar{f}(u)=\frac{\sum_v f^0(v)}{n}$ を各要素とする $m$ 次元ベクトルである. また, Lagrange 関数

$$Lag(z,d)=\frac{1}{2}z^TW^{-1}z+(b-Bz)^Td,$$

に対する最適性の十分条件 $\nabla_z Lag = W^{-1}z-B^Td=0$ より, 解は

$$\mathbf{z}=WB^T\mathbf{d},$$

$$L\mathbf{d}=\mathbf{b}. \tag{7}$$

を満たす（$d$ は Lagrange 乗数）.

上記の形式から, (1) を解くことは, 負荷の均衡条件 (6) を満たす可能解のうち, 重み付き $l_2$-ノルム (5) を最小化するものを求めること, 言いかえれば, （自分自身に負荷を戻す）巡回フローによる無駄な負荷移動を避けた均一解を求めることを意味する. ただ, グラフが巡回路を持つ時は, いかなる計算スキームを用いても本質的に (5)(6) と等価な問題を解くことになるので, 大域変数を扱えない分散システム上の局所処理では特に反復計算が必要になるものと思われる.

## 3.2 全域木上の効率的なメッセージ伝搬

前節の問題点を逆手にとって, 巡回路を持たない全域木で予めネットワークを構成しておけば, 葉-根間のメッセージ伝搬により**直接的に負荷移動量が求められる**. 具体的には, 拡散法 (1) では任意の時刻で総負荷量 $\sum_{v\in V}f(v)$ が一定値に保存される [6]

ことを利用して, 開始時に各サーバの負荷量を根に送付累積させ, 根で計算した均一負荷量 $\bar{f}$ をブロードキャストした後, $\bar{f}$ と各サーバの負荷量 $f(u)$ との差 $z_e = f(u) - \bar{f}$ を葉から順に求めていけば良い（親の頂点 $v$ では子等に移動する分を考慮して $z_{e'} = f(v) - \bar{f} + \sum_e z_e$）.

また, 各辺 $e \in E$ 上の移動量 $z_e$ がわかれば, **葉から順に親に対して** $z_e$ だけの負荷の要求あるいは供給を**非同期に出来るところから実行**していけば, 移動順による負荷の過不足や無駄な移動の問題は起らない. この計算量は $O(m)$ である. もちろん, 近傍に供給すべき量だけ負荷が伝搬移動してくるまでの遅延は生じるが, （新たなバイパス経路による移動を考えない限り）これは初期配分 $\mathbf{f}^0$ に依存した必要不可欠な移動なので, どんなアルゴリズムを用いても本質的に避けられない. もちろん, ボトルネックとなる遅い経路やサーバが出来るだけ少なくなるように全域木を構成することが有効である.

# 4 まとめ

分散ネットワーク上のサーバの負荷均一化問題を考え, その結合特徴を反映した Laplacian による拡散法が, 多項式による反復, 連立方程式, 2 次計画問題などのさまざまな形式に帰着することを整理した. また, 効率的な処理を行うには, 負荷の均衡のみならず無駄な巡回フローを抑えることが本質的に重要であることを指摘した. 一方, 巡回路を持たない全域木を考えれば, 従来の計算スキームのような反復処理をする必要がなく, メッセージ伝搬によって直接的に負荷移動量が求められるとともに, 分散システムに適した非同期な負荷移動が効率的に実現できることを示した.

このような全域木上の迅速な処理は, 実行中に負荷量自身が変動する場合にも適応的と考えられる. また, 拡散法における最適なトポロジーは未知 [3] であるが, 分散ネットワーク環境では柔軟に結合（や経路）が変化する方がむしろ自然で, 必要に応じて全域木を構成し直すのが現実的なのかも知れない. そこで今後は, 巡回路を持つネットワークを全域木で近似した高速解法や, 各サーバの性能などが異なるヘテロなネットワークへのアルゴリズムの拡張を中心に検討を深めていきたい.

# 参考文献


[1] F.R.K. Chung, Spectral Graph Theory, American Mathematical Society, 1994.

[2] G. Cybenko, "Dynamic Load Balancing for Distributed Memory Multiprocessors," *Journal of Parallel and Distributed Computing*, Vol. 7, pp. 279-301, 1989.

[3] T. Decker, B. Monien, and R. Preis, "Towards Optimal Load Balancing Topologies," A. Bode et al. (Eds): Euro-Par2000, *LNCS* 1900, pp. 277-287, 2000.

[4] R. Diekmann, A. Frommer, and B. Monien, "Efficient Schemes for Nearest Neighbor Load Balancing," *Parallel Computing*, Vol. 25, pp. 789-812, 1999.

[5] R. Elsässer, A. Formmer, B. Monien, and R. Preis, "Optimal and Alternating-Direction Load Balancing Schemes," P. Amestoy et al. (Eds.) Euro-Par'99, *LNCS*, 1685, pp. 280-290, 1999.

[6] 林 幸雄. "グラフの Laplace-Beltrami 作用素とその応用," **数理解析研究所講究録**, No. 1241, pp. 39-47, 2001.

[7] Y.F. Hu, R.J. Blake, "An Improved Diffusion Algorithm for Dynamic Load Balancing," *Parallel Computing*, Vol. 25, pp. 417-444, 1999.

[8] 仁木 滉, 河野 敏行. **楽しい反復解法**, 共立出版, 1998.

[9] V.S. Sunderam, and G.A. Geist, "Heterogeneous Parallel and Distributed Computing," *Parallel Computing*, Vol. 25, pp. 1699-1721, 1999.

[10] R.S. Varga （渋谷 他訳）. **計算機による大型行列の反復解法**, サイエンス社, 1972.

[11] C. Xu, and F.C.M. Lau, Load Balancing in Parallel Computers -Theory and Practice-, Kluwer Academic Publishers, 1997.