社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－DPS－137 (11)
2008／11／28

P2Pネットワーク上での類似度検索のためのデータ配置方法

鈴木伸和† 菅谷至寛† 阿曽弘具†

膨大な数のコンテンツを有する分散型データベースにおいて，効率的な検索手法が重要となる．その中で，データの類似性をもとに内容の近いものを自動的に得ることができる類似度検索は必要な情報を取得する上で有効である．そこで，本稿ではP2Pネットワーク上で効率的な類似度検索を行うためのデータ探索手法を提案する．インデックスをcounting filterもしくはMDSによって多次元多値ベクトル化し，構造化オーバーレイを用いた類似度検索システムのアーキテクチャを与える．シミュレーション実験の結果，ルーティングコストは$O(\log N)$となり，counting filterを用いた手法が，MDSを用いた手法よりも精度が高いことが確認された．

Data Distribution for Similarity Retrieval on Peer-to-Peer Network

Nobukazu Suzuki,† Yoshihiro Sugaya† and Hirotomo Aso†

An efficient search method is required on a distributed database which has large amounts of data. Especially, Similarity Search is one of the important methods that enables efficient and high quality search in peer-to-peer database. In this paper, we approach the data distribution and search method for efficient similarity retrieval on decentralized and distributed network. Our method uses counting filter or multidimensional scaling for indexing, and exploits structured overlay network. We evaluate the cost of searching and the recall, and show that routing cost is $O(\log N)$, and the accuracy of counting filter approach is higher than that of MDS.

1. はじめに

近年，インターネットの普及により，Web上に存在する情報量は増加の一途を辿っている．膨大な情報の中から有益な情報を見つけ出すために，インターネットを利用した情報検索システムや情報共有システムが注目されている．本研究では，データセット全体から，あるデータに類似したデータを探すという類似度検索に着目し，P2Pネットワーク技術を利用したデータベース上での類似度検索システムのアーキテクチャを与える．

P2Pシステムが構築するネットワークは，非構造化オーバーレイと構造化オーバーレイに大別される．構造化オーバーレイはノード間の隣接関係がルーティングアルゴリズムによって制約されたネットワークであり，近年盛んに研究されている．代表的な構造化オーバーレイとしてChord[1]などの分散ハッシュテーブル（DHT）が挙げられる．DHTによる検索効率はノード数Nに対して$O(\log N)$となるが，一致検索が対象で，範囲検索を対象としていない．範囲検索が可能である構造化オーバーレイの一つに，Skip Graph[2]がある．Skip Graphは，DHTと同様に検索コストが$O(\log N)$である．Skip Graphはキーとノードが1対1に対応しているが，単一ノードが複数のキーを保持可能であるような拡張されたSkip Graphが提案されている[3]．

以上のいずれも検索のためのデータ表現として比較的低次元のベクトルを対象としている．コンテンツの類似性を比較するためのデータ表現は，素直な形では非常に高次元となる．そこで，本研究では，分散データベースにおいて高次元ベクトルに対する範囲検索を実現するため，低次元化の方法を二つ提案し，ルーティングコストや検索メッセージ数を小さくし，高精度な検索を可能とするP2P検索システムを提案する．提案システムの評価実験を行い，その性能を確認した．ルーティングコストや検索メッセージ数を小さくするためにデータの分散配置の方法が重要で，低次元化の方法はその効率的配置法の組織的方法を与えている．範囲検索を可能とする低次元データの最終的配置には拡張したZNetを用いた．

2. 大規模分散データベース

データベースの中から適切な情報を探し出すための方法として，類似度検索がある．ほしい情報を表すク

† 東北大学 大学院 工学研究科 電気・通信工学専攻
Department of Electrical and Communication Engineering, Graduate School of Engineering, Tohoku University

エリデータを入力して，それに類似したデータを抽出するものである．データの類似性をデータがもつ情報の類似性として計る必要があるため，データがもつ情報を適切に表現しなければならない．データがもつ情報をコンテンツと呼ぶ．本研究では，コンテンツの表現としてキーワードの集合を考える．多数のキーワードを想定しておき，コンテンツを特徴付けるキーワードを選び，その集合をインデックスデータ，あるいは単にインデックスと呼ぶ．インデックスデータ間にはコンテンツの類似性を反映した類似度が定義できるものとする．これをデータの類似度と呼ぶ．

このようなインデックスを利用した情報共有サービスの例として，「本棚.org[4]」がある．「本棚」と呼ばれるユーザーの蔵書セットを登録して公開することができるサービスで，本棚は書名をキーワードとしたインデックスと言える．異なる本棚間で共通して現れる書籍の割合を類似度として，ある本棚に類似した本棚のリストを得る類似度検索が可能になっている．

普遍集合が想定できるときその部分集合は 0,1 を要素値とする特性ベクトルとして表現できる．全キーワードを想定すれば，インデックスの特性ベクトル表現が可能で，それをデータの特徴ベクトルと呼ぶ．インデックスデータを P2P ネットワーク上で分散管理するデータベースを考える．クエリデータに対する類似度が大きいデータから順に k 個を列挙する top-k 検索システムを設計することが目的である．無駄な検索を避けるため，インデックスデータを適切に配置することが重要となる．

2.1 インデックス分散の方法

類似度検索では，あるデータに対し，類似度が上位のデータだけが重要となり，全く類似しないデータにアクセスする必要はない．しかし，類似するかどうかは類似度を計算してみなければ分からない．そこで，データの類似性を反映したデータ配置を定め，位置が遠ければ類似性が低くなるようにすることができれば，類似しないデータへのアクセスを不要にできる．すなわち，分散環境において，類似コンテンツが近くのノードに配置されるようデータ配置の局所性を高めることにより，効率的な類似度検索が実現できる．コンテンツを非常に低次元な特徴空間で表す事ができれば， ZNet のような多次元での範囲検索が可能な構造化オーバーレイを類似度検索のために利用することができる．

しかし，本研究で対象としているデータの特徴ベクトルは非常に高次元かつスパースであることが多い．本棚.org を例に取れば，書籍のリストは (本 1, 本 2, ..., 本 n) のような非常に長くスパースな 2 値ベクトルとして表現される．各要素は当該リストでのそれぞれの本の有無を表し，ベクトル長はデータベース全体での本の種類数となり非常に大きい．ZNet では，多次元ベクトルを空間充填曲線によって 1 次元に変換する方法を用いており，元の空間が低次元多値ベクトルで表現されていれば局所性をある程度保てるが，高次元では距離関係が保てなくなる．Web ページに対する検索エンジンなどでは，各ページに現れる単語に対して重み（tf-idf 値など）を計算し，重要な単語を抽出してより低次元の特徴ベクトルを作成して検索を行う手法が用いられる．インデックスの空間は非常に高次元でスパースであるため，既存の手法をインデックス集合に用いたとしても，有効な次元削減を得ることは難しい．また，中央集中型の検索エンジンで用いられる手法は，信頼できる結果を得ることができる一方で，全データの情報を必要とする．そのため， P2P アーキテクチャで同様のシステムを実現するには，ネットワーク内を網羅するような大域的な情報を必要とし，総じて維持コストが高くなってしまう．

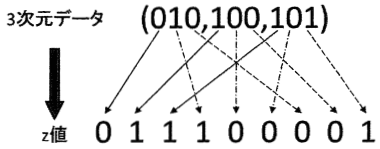

図 1 インターリーブ処理の例（$d = 3$, $t = 3$）

そこで，大域的な情報をできるだけ必要としない方法でインデックスデータをデータ間の局所性をできるだけ保ったまま低次元の多値ベクトルへ変換する手法を用い，かつ高い精度が得られる P2P 検索システムの構築を目指す．

3. 構造化 P2P システム:ZNet

本章では，多次元ベクトルを扱うことのできる構造化オーバーレイである ZNet について説明する．

3.1 ZNet の概要

多次元ベクトルを局所性を保存したままマッピングする構造化オーバーレイに，ZNet がある．ZNet では多次元ベクトルを空間充填曲線である Z-ordering によって 1 次元（Z 値）に変換する．各次元が 0 から $2^t - 1$ の整数値をとる多次元ベクトル $(x_1, x_2, ..., x_d)$ が与えられ，次元 i の値 x_i を t 桁の 2 進数で表現すると，Z 値はビットインターリーブ処理[5] によって dt 桁の 2 進数として求まる．図 1 にインターリーブ処理の例を示す．多次元ベクトルをその Z 値に応じた位置に配置することによって，はある程度局所性を保ったまま 1 次元数直線上に配置される．

ZNet では，ネットワークのトポロジとして通常の Skip Graph と同様にしてネットワークの参加時に割り当てられた membership vector に基づいてオーバーレイを構築する． Skip Graph の構造を図 2 に示す．ZNet は Skip Graph を拡張し，複数のキーを保持できるように各ノードは Z 値に関して連続した領域

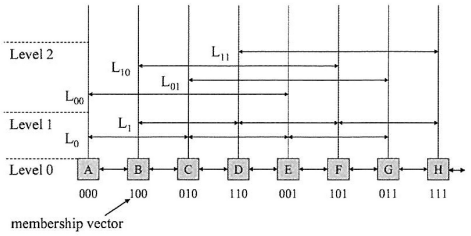

図 2 Skip Graph の構造

表 1 ルーティングテーブル（ノード D の例）

Level	Neighbor Node		Z Range
2	lN	-	-
	rN	H	[192,255]
1	lN	B	[128,159]
	rN	F	[172,191]
0	lN	C	[160,163]
	rN	E	[168,171]

が割り当てられる．これを Z Range と呼び，各ノードの Z Range は割り当てられた領域の Z 値の最小値 zMin 及び最大値 zMax を用いて [zMin,zMax] と表すことができる．隣接ノードはレベル毎に左右の 2 つ (lN, rN) が定義され，レベル i では membership vector の上位 i 桁が同じノードを隣接ノードと定める．レベル 0 で隣接するノードは，管理する Z Range も連続している．各ノードは表 1 のようなルーティングテーブルを管理する．ルーティングテーブルには各レベルの隣接ノードと隣接ノードが管理する Z Range を格納する．

3.2 検　　索

ZNet は Skip Graph と同様に，完全一致検索も可能である．DHT と同様に put (key,value) 操作と get (key) 操作によってルーティングが行われ，クエリとなるデータを格納するノードを探索する．

多次元データに関する範囲検索は，RangeSearch[3] アルゴリズムによって行われる．多次元空間上で矩形領域を指定し，この領域の各点の Z 値を求める．これらの値の集合は，複数の Z Range に分割され，それぞれの Z Range を管理するノードをルーティングによって探索する．

4. 提 案 手 法

データの局所性を保存でき，多次元ベクトルを扱える ZNet で，範囲検索を利用して類似度検索を行うためのインデックス配置方法を提案する．2. で述べたように，インデックスをそのままベクトルとして表現すると，高次元 2 値ベクトルとなる．そのため，Z-ordering に従ってインデックスを配置しても，インデックスの局所性を反映することができない．そこで，

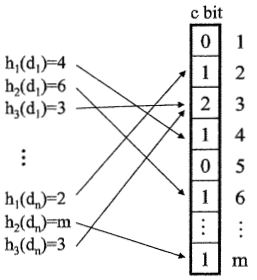

図 3 counting filter の作成

局所性を保つようなインデックスの低次元多値ベクトル化の手法として，counting filter を用いる方法と，MDS（multidimensional scaling）を用いる方法を考案した．

4.1 Counting filter を用いたデータ配置

過去に Bloom filter[6] を利用した非構造化オーバーレイ上での類似度検索システムについて提案を行った[7]．ここでは，Bloom filter の拡張である counting filter を利用した構造化オーバーレイでのデータ分散手法について述べる．

4.1.1 Counting filter

counting filter は Bloom filter のカウンタをビットからビット列に拡張したものである．counting filter はハッシュ関数値を添字番号とする配列である．

counting filter は図 3 のように，m 個の c ビットカウンタからなり，初期状態は全ての数値が 0 である．$[1, m]$ の値をとる k 個の異なるハッシュ関数によって格納したいキーワード d_i のハッシュ値 $h_1(d_i), ..., h_k(d_i)$ (値は 1〜m) を求め，ハッシュ値と番号が一致するカウンタに 1 を加える．図 3 は $k = 3$ の場合の，キーワードの集合 $d = \{d_1, ..., d_n\}$ に対する counting filter を表している．例えば，図 3 において $h_1(d_1) = 4$, $h_2(d_1) = 6$, $h_3(d_1) = 3$ であるので，4 番目，6 番目，3 番目のカウンタにそれぞれ 1 を加える．キーワードの集合（インデックスデータ）d から作られた counting filter を CF (d) とし，その m 次元ベクトルを CF 値と呼ぶ．あるキーワード key_q が d に存在するかどうかをチェックしたいときは，格納に用いた k 個のハッシュ関数にキーを通して指定された番号で，CF(d) の値が全て 1 以上であるかをチェックすればよい．そのチェックが通った場合，そのキーを保持してる可能性があると分かり，通らない場合はそのキーは確実にないことが分かる．新しくキーを挿入 (削除) するときは，ハッシュ関数によって指定されたカウンタの値に 1 加える (減ずる)．キーの追加・挿入・チェック操作の計算量はいずれも $O(k)$ である．

4.1.2 インデックス配置と検索

counting filter では，各キーワードに対応するカウンタがハッシュ関数によって一意に定まる．そのため，

各カウンタはハッシュ関数を用いて選ばれたキーワードのサブセットに対応するカウンタということになる．その結果，要素として含んでいるキーワードの構成が似ているインデックス同士であるほど，カウンタの値は近い値になる．

インデックスの配置は以下の様にして行われる．

- インデックスの CF 値を計算
- CF 値からインターリーブ処理によって Z 値 z を計算
- get(z) によって発見したノードにインデックスを転送配置

検索操作は，インデックスの CF 値を計算し，CF 値を中心とし，各次元の検索幅を r とした矩形領域の範囲検索により行う．目的はインデックスの類似度指標において，top-k 検索を行うことであるから，検索領域を管理するノードにクエリが届いたら，インデックスの類似度計算によってクエリインデックスと類似したインデックス k 個をローカルデータから取得して，情報を転送する．r ははじめ狭く設定しておき，満足した結果が得られないときは r を大きくして再検索を行う．

4.2 MDS を用いたデータ配置

次に，MDS を用いてデータ間の距離行列から多次元ベクトルを定義する方法について述べる．

4.2.1 MDS

MDS は観測された対象間の非類似度 (距離) が与えられたとき対象を多次元空間内の点として表し，点間のユークリッド距離と観測された非類似度が数値的に最も良く一致するように点の空間配置を定める方法である[8)]．対象としているデータ $o_i(i=1,...,n)$ に対して s 次元空間内の座標 $\mathbf{x}_i$ で，$\|\mathbf{x}_i-\mathbf{x}_j\|=d_{ij}$ がデータ間距離 $d(o_i,o_j)$ の関係をできるだけ満たすものを求めたい．この解法の 1 つに，$\{x_i\}$ の重心を原点に対応させるものがある．

この方法の計算の概略を示す．与えられたデータ集合 $\{o_1,o_2,...,o_n\}$ に対応するベクトルの集合 $\{\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_n\}$ の平均ベクトルを $\bar{\mathbf{x}}$ とおく．$b_{ij}=(\mathbf{x}_i-\bar{\mathbf{x}})(\mathbf{x}_j-\bar{\mathbf{x}})$ を (ij) 要素とする $n\times n$ 行列を $\mathbf{B}$ とおく．このとき，$\mathbf{x}_i^T\mathbf{x}_j=\frac{1}{2}(\|\mathbf{x}_i\|^2+\|\mathbf{x}_i\|^2-d_{ij}^2)$ $(d_{ij}=\|\mathbf{x}_i-\mathbf{x}_i\|)$ の関係を用いて，次の式が得られる．

$$b_{ij}=\frac{1}{2}\Big(\frac{1}{N}\sum_{k=1}^{N}d_{ik}^2+\frac{1}{N}\sum_{k=1}^{N}d_{kj}^2 -\frac{1}{N^2}\sum_{k=1}^{N}\sum_{l=1}^{N}d_{kl}^2-d_{ij}^2\Big). \qquad (1)$$

この式はデータ間距離だけを用いて表されており，データ間距離を求めると行列 $\mathbf{B}$ を求めることができる．この $\mathbf{B}$ をスペクトル分解することにより，データ a_i に対応するベクトルとして $\mathbf{x}_i-\bar{\mathbf{x}}$ が求まり，$\mathbf{x}_i$ が決定できる（$\{\mathbf{x}_i-\bar{\mathbf{x}}\}$ の平均がゼロベクトルなので，$\bar{\mathbf{x}}=0$ とおける）．

$\mathbf{B}$ の非ゼロ固有値の個数が s の値を与える（小さい値を無視する事で，より低次元の近似解も可能）．こうして得られた対応を MDS マップと呼ぶ．MDS の計算量は $O(n^3)$（n はデータ数）となる．

4.2.2 MDS マップの統合

正確な MDS 座標を求めるには，データを集中管理して，全てのデータを元に MDS の計算を行うサーバが必要となるが，これはスケーラビリティに乏しい．また，集中管理型であれば，MDS マップ上での座標はデータ間の距離関係が保たれているので類似度検索に影響はない．しかし，分散データベースでは分散性の有効利用のため，全データ集合をデータ 群に分け，それぞれで MDS 座標を求める．従って，異なったデータ群に対して求まる MDS 座標系は異なるものであり，互いのベクトルを比較することができない．すなわち，MDS 座標系が一致していなければ類似しているはずのデータがネットワークにおける近傍に集まらない．そこで，Shang らの手法[9)] を用いて，隣接するノードが持つデータとローカルデータからなる MDS マップを，各レベルのリンクに対して作成し，これらをひとつのマップに統合することによって，グローバルな MDS 座標を決定する．

(1) 2 つの MDS マップ M，M' 間で同一のデータに対応する点集合をそれぞれ M_I，M_I' とする

(2) M_I と $T(M_I')$ 平均の誤差率を最小にするようなアフィン変換 T を求める

(3) データ P に当たる座標 p を次のようなルールで決定し新しいマップを生成する
- 点 p が M にあって M' にない場合，M の座標を用いる
- 点 p が M' にあって M にない場合，$T(M')$ の座標を用いる
- 点 p が M と M' の両方にある場合，M と $T(M')$ の座標を平均したものを用いる

(2) で $T(M')$ を求める手順は具体的に以下のようになる．

- マップ間で共通して現れるデータをランダムに $s+1$ 個選び，M，M' での座標をそれぞれ $\mathbf{x}_c^i$，$\mathbf{x}'^i_c$ $(i=1,...,s+1)$ とする．
- $\mathbf{x}_c^i$ は $\mathbf{x}'^i_c$ をアフィン変換することによって求められると仮定し，$\mathbf{A}=\{a_{ij}\}$，$b=[b_1,...,b_s]^T$ $(i,j=1,...,s)$ として，アフィン変換 $\mathbf{x}_c^i=\mathbf{A}\mathbf{x}'^i_c+\mathbf{b}$ に $\mathbf{x}_c^i$ および $\mathbf{x}'^i_c$ を代入して得られる a_{ij}，b_i についての連立方程式として解き，アフィン変換を求める
- M' 上の各点 $\mathbf{x}'_i$ を上記で求めた $\mathbf{A}$ と $\mathbf{b}$ を用いてアフィン変換した座標を $T(M')$ とする

4.2.3 MDS マップの統合とデータの再配置

ZNet におけるレベル 0〜i-1 のリンクにおいて以下の操作を行い，グローバルな MDS マップ GMDS を作成する．

(1) 隣接ノードのデータとローカルデータとでMDSマップを作成
(2) ひとつ下のレベルで作成したMDSマップとマージ
(3) マージして得られたマップを元に，各ノードが持つデータのMDS座標を更新

GMDSを作成したのち，ローカルデータの中で自身のZ-rangeに含まれないGDS座標は，get(GMDS座標)操作によって，GMDS座標に対応するデータを保持するノードをネットワーク内から探し，発見したノードにデータを転送する．実験では，MDSマップの統合とデータ転送の操作を各ノードが複数回繰り返すことによって，MDS座標の更新を行ってGMDS座標を得ている．

4.2.4 検　索

検索のために，dt 桁の値を出力するハッシュ関数Hashを用いて，各ノードは保持するデータのIDに関するハッシュ値Hash(Data ID)を計算し，Hash(Data ID)を満たすZrangeに含むノードが(key,value)=(Hash(Data ID),GMDS座標)というタプルを保持する．検索クエリは，インデックスのIDと検索範囲 r_q からなる．Hash(Data ID)をZrangeに持つノードからGMDS座標sを取得し，各次元の検索範囲を $[s_i - r_q, s_i + r_q]$ とした矩形領域を検索する．目的はインデックスの類似度指標において，top-k検索を行うことであるから，counting filterの検索方法と同様に，検索領域を管理するノードにクエリが届いたら，インデックスの類似度計算によってクエリインデックスと類似したインデックス k 個をローカルデータから取得して情報を転送する．

4.3 計 算 量

インデックスから多次元データの座標を求めるときの計算量は，counting filterでは $O(lk)$（l はインデックス内のキーワード数）である．MDSでは，まず各ノードがローカルに保持するデータの距離行列の作成には $O(n^2)$（n は1つのノードが持つデータ数）かかり，距離行列から座標を算出するために $O(n^3)$ かかる．

ルーティングコストやメッセージ数はSkip Graphに従う．すなわち，ルーティングコスト，メッセージ数は $O(\log N)$ となる．よって，各ノードでGMDSの作成にかかるコストは $O(n^3 \log N)$ となる．

5. 実　験

本節では，提案したデータ配置方法の性能を評価する．

5.1 実験環境

シミュレーションにおけるデータセットには4741人の所有書籍リストを用い，データ間の類似度は，ダイス係数及びコサイン係数を用いた．データ I，I' に現れるキーワード数をそれぞれ $|I|$，$|I'|$ とし，I と I' のどちらにも現れるキーワード数を $|I \cap I'|$ としたとき，ダイス係数は $\frac{2|I \cap I'|}{|I|+|I'|}$ と表され，またコサイン係数は $\frac{|I \cap I'|}{|I||I'|}$ と表される．

表2 実験パラメータ

試行回数	10
ネットワーク	
ノード数	N=256,512,768,1024
インデックス数	4741
全キーワード数	16,0000
Counting filter	
カウンタのビット数	c=10
カウンタ数	m=16
ハッシュ関数の数	k=3
MDS	
共通データ数	C=100
MDS空間の次元数	s=16
GMDSの作成回数	g=5
検索で求めたいデータ数	k=10
検索クエリ候補数	500

データセットのうち，およそ10%（500個）を検索クエリ候補とする．counting filterを用いる手法ではインデックスから計算されるcounting filterを求め，Z値に従ってZNet上に配置する．MDSを用いる手法では，検索クエリ候補を除くデータからランダムに100個（全体の2%）をMDSマップを作成するための全ノードが持つ共通データセットとし，残りのデータをランダムに分散させる．GMDSの作成とデータ転送を全体で g 回繰り返し，データの再配置を行ってから検索を開始した．シミュレーション環境はJava 1.6.0で実装し，Core 2 Duo 2.1GHz，RAM 6GB，OS Debian GNU/Linux 4.0（カーネル2.6.18 amd64）のマシン上で行った．

評価した内容は範囲検索のルーティングコストとメッセージ数および，範囲検索におけるヒット率である．それぞれ，10回試行した平均を求める．

5.1.1 範囲検索のルーティングコストとメッセージ数

一回の範囲検索でクエリメッセージが経由するノード数（経由ノード数）をルーティングコストとし，提案手法における範囲検索のルーティングコストとメッセージ数の平均を図4に示す．用いる類似度の違いが現れなかったため，図では双方の類似度を区別せず平均を算出した．検索範囲は $r \in [0.05, 0.25]$ とした．

どちらの手法でも1回の検索で経由ノード数は平均10〜20ノードであり，ノード数の増加に従って対数的に増加することが分かる．一方，検索によって発生するメッセージ数は平均30〜50で，経由ノード数の3倍ほどとなった．メッセージ数もルーティングコスト同様，ノード数の増加に従い対数的な増加を示している．

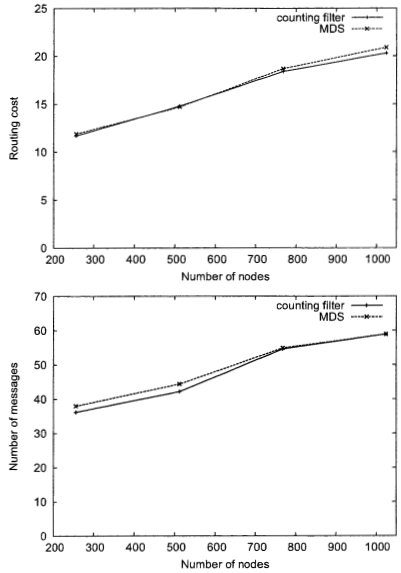

図 4 （上）ルーティングコスト（下）メッセージ数

5.1.2 範囲検索の精度

検索範囲 r を $0-100$ の間で変化させたときに得られる top-k 検索結果の精度を評価する．精度の定義は以下に示す top-k 検索の結果に含まれる正解データ数とし，$\frac{|D_{relevant}|\cdot|D_{retrieved}|}{k} \times 100[\%]$ を計算して求まる．ただし，$D_{relevant}$ は非分散環境で得られる top-k の正解データの集合，$D_{retrieved}$ は提案手法を用いた検索によって得られたデータ集合である．k は top-k 検索で上位 k 個までを求める事を意味する．

結果を図 5 に示す．検索範囲 r を大きくすればするほど精度は上がるが，$r = 100$ でも counting filter を用いたときの精度は 66%，MDS を用いたときの精度は 63%と低いことから，類似したデータの一部は多次元データ空間上で離れた位置に配置されてしまっていることが分かる．また，counting filter を用いた手法が MDS を用いた手法よりも総じて精度が高いことが分かる．実験では，Dice 係数と Cosine 係数という 2 つの異なる類似度指標を用いたが，その差は現れなかった．

6. ま と め

本論文では，構造化型類似度検索システムを提案した．インデックスを多次元データに変換し，多次元データを扱える構造化オーバーレイ ZNet によって範囲検索を行う．多次元データ化には 2 つの手法を用いた．シミュレーション評価により．ルーティングコストはネットワークサイズに対して，対数的な増加で済む事を示した．ルーティングコストは低く抑えられたといえる．検索範囲を拡大することによって精度が上昇するが，より局所性を考慮したデータ表現を用いることによって，精度の改善が見込める．今後は，インデックスのよりよい低次元多値ベクトル化の手法を実現することと，データ複製や，検索結果に基づくルーティングの変更機構を付加し，動的に最適化を行う，効率的かつ高精度なシステムの構築を目指したい．

図 5 検索範囲 r と精度の関係

参 考 文 献

1) I. Stoica, R. Morris, D. Karger, M. F. Kaashoek and H. Balakrishnan : “Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications,” Proceedings of ACM SIGCOMM 2001, pp.149-160, Aug. 2001
2) J. Aspnes and G. Shah, “Skip Graphs,” Proceedings of SODA ’03, pp. 384-393, Society for Industrial and Applied Mathematics, 2003.
3) Y. Shu, B. C. Ooi, K.-L. Tan, and a. Zhou, “Supporting multi-dimensional range queries in peerto-peer systems,” Proceedings of P2P’05, vol. 00, pp. 173-180, IEEE Computer Society, 31 Aug.-2 Sept. 2005.
4) 本棚.org， `http://hondana.org/`.
5) R. Bayer, “The universal B-Tree for multidimensional Indexing,” Technical Report TUM-I9637, Institut fur Informatik, TU Munchen, 1996
6) B. Bloom. “Space/time trade-offs in hash coding with allowable errors,” Comm. of ACM, pages 13(7):422-426, July 1970.
7) 鈴木 伸和，菅谷 至寛，阿曽 弘具，“P2P ネットワーク上での効率的な類似度検索，” 情報処理学会研究報告，2008-DPS-134, Vol.2008, No.21, pp.225-230 (2008) .
8) 齋藤 堯幸，宿久 洋，“関連性データの解析法——多次元尺度構成法とクラスター分析法，” 共立出版， 2006
9) Y. Shang, W. Ruml, “Improved MDS-Based localization,” Proceedings of Infocom 2004, vol. 4, pp. 2640-2651, March 7-11, 2004.