D-005

空間 Web データにおける m-最近接キーワード検索方式 DCC の性能評価
Evaluation of m-CK search algorithm DCC on web spatial data

邱 原 † 大森 匡 † 新谷 隆彦 † 藤田 秀之 †

Yuan Qiu Tadashi Ohmori Takahiko Shintani Hideyuki Fujita

1. はじめに

近年，Flickr や Twitter のような空間的な位置情報を持つ空間 Web データが増えている．空間 Web データの利用方法の 1 つに，キーワード m 個の入力を受け，当該キーワードを満たす高々m 個のオブジェクト (Web データに応じるオブジェクト) 群のうち最も近接している組を探す問題 (m-CK 検索)[1] がある．

この問題に対して，各キーワードに関連するオブジェクト集合を組合わせてから，最小直径を持つオブジェクト集合を探すという方針を沿って，組合わせる方法や枝刈りルールなどを工夫しながら，実用性が高いデータ構造を選べることも重要である．

先行研究として，Zhang らは事前に全データを 1 つの R-tree で保存することを前提に，Apriori で R-tree の MBR の集合を列挙する方法を提案した [1].

しかし，Twitter や Flickr などのサーバ側では Grid 分割した階層データ構造によるデータ管理しか期待できない．そこで我々は，Grid データ構造でデータを保存して，Grid のセルにキーワードに応じた MBR を用意し，各キーワードのセルを 1 ノードとして選び，そこからサイズ m のノードセットを作って，その列挙の優先順を制御する探索方式 DCC(Diameter Candidate Check) を提案した [2].

さらに，現実的には，サーバ側が単純にインデクスの無いデータ集合しか提供しない場合も考えられる．そのため，本稿は事前に Grid データ構造を用意せず，問合せ発生の時に On-demand で Grid を生成してから mCK 検索を行う方式を述べる．次にノード間の距離の下限を利用してノード集合の内検索対象にならないノードを削除する枝刈り規則を追加する．本稿では，以上の条件下で DCC 方式の評価を行い，有効性を示す．

2.On-demand による Grid の生成

本稿は問合せ時に必要なデータに対して，キーワードごとに Grid 構造を作る方針でデータを取り扱う．図 1 によって，問い合わせる時，クエリのキーワードごとに，Flickr や Twitter のキャッシュデータ集合から各キーワードに関連するデータをロードし，それぞれの Grid 構造を用意しておく．そして，各 Grid のセルを 1 ノードとしてサイズ m のノードセットを組み合わせて，mCK 検索をする．ただし，Grid のセルには MBR 情報をつけている．キーワード別に Grid 構造を生成すると，データをロードするためのコストが増加するが，単一 Grid の場合よりもデータの数が少なく，深さも小さくなるため，検索時ノードの組み合わせ数を減少する．

†電気通信大学, UEC

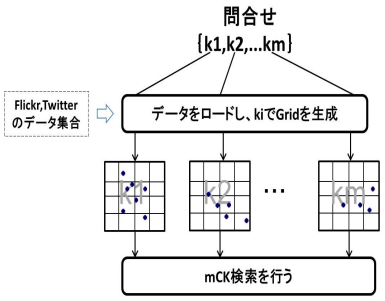
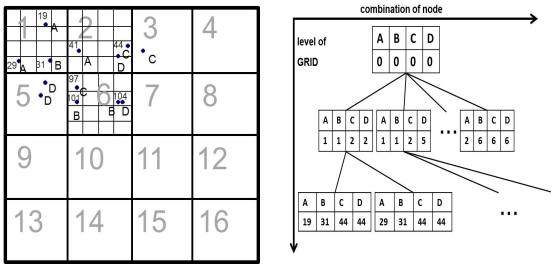


図 1: On-demand で Grid 生成

3.DCC の概要

データ分布の例と探索空間木は図 2 に示す．$\{A, B, C, D\}$ はキーワードであり，数字は Grid 上のノード番号を表す．記号 $A[i]$ はキーワード A に関連する番号 i のノードを示す．各キーワードに関連するノードの組み合わせ $\{A[i], B[j], C[k], D[l]\}(i, j, k, l$ はノード番号である) をノードセットと呼ぶ．探索空間木に対して，深さ優先で展開しながら，先に計算したノードセットの結果を閾値としてその後のノードセットの枝刈りを決めるということである．探索順番つまり組み合わせるノードセットの優先順は探索効率に大きい影響を与える．そのため，より小さい直径を先にもとめると，探索効率がよいので，我々は [2] で DCC(Diameter Candidate Check) 方式を提案した．



(a) データ分布 (b) 探索空間木

図 2: mCK の例

オブジェクト集合の近さは，任意の 2 オブジェクト間の距離の最大値 (**直径**) によって決まる．つまりオブジェクト集合は全部の m 個のオブジェクトを使用せず，二つのオブジェクトだけ用いても直径の候補は判る．そこで，二つのオブジェクトから構成されるペアの集合を作って，小さい順で各ペアにおいて直径としたオブジェクト集合を組み立てることができるかどうかを判定する方が効率がよい．DCC の流れは図 3 の 3 ステッ

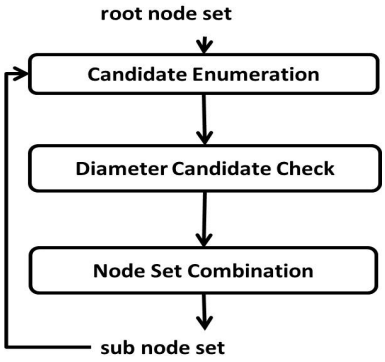


図 3: DCC の流れ

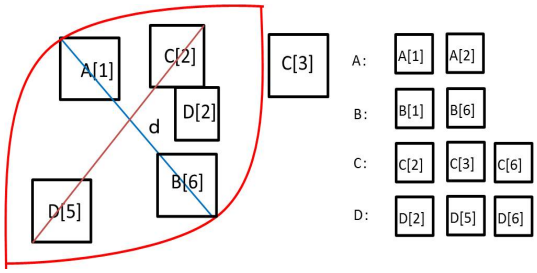


図 4: DCC の例

プで表される.

[Step1:Candidate Enumeration] Diameter Candidate(DC) の集合を作ることである. 異なる 2 キーワードに関連するノード集合のうち，それぞれ 1 ノードを選び Diameter Candidate とする. 例:図 4 各キーワードに関連するノードについて，異なるキーワードの組 A と B，A と C，…，C と D に関連するノードを組み合わせると，$\{< A[1], B[1] >, < A[1], B[6] >, < A[1], C[2] >, ..., < C[6], D[6] >\}$ 計 37 の Diameter Candidate とする．全部の DC をノード間の最大距離によってソーティングをする.

もしある DC のノード間の最小距離が閾値より小さい場合，枝刈りできると判定する.

[Step2:Diameter Candidate Check] Diameter Candidate Check はある Diameter Candidate に対して，各ノードが Diameter Candidate の近傍にあるかどうかを判定するということである. 2 ノード $\{n_i, n_j\}$ の MBR 間の最大距離を $Maxdistance(n_i, n_j)$ と表記する．例えば,Diameter Candidate$< A[1], B[6] >$ に対してノード $C[2]$ をチェックする場合
$Maxdistance(A[1], C[2]) < Maxdistance(A[1], B[6])$ かつ
$Maxdistance(B[6], C[2]) < Maxdistance(A[1], B[6])$ を満足すれば,$C[2]$ をノードセットの候補として残す. 同じように，$D[2], D[5]$ を残すが，$C[3]$ が残さない (図 4).

もしあるキーワードのノードが 1 つも残っていないなら，該当 Diameter Candidate からノードセットを組み合わせられないのでその DC は枝刈りされる.

[step3:Node-Set Combination] 各キーワードについて残っているノードを nested loop で組み合わせしながら，ノードの相互間の最大距離と Diameter Candidate の最大距離と比較によって枝刈りを行う. 組み合わせたノードセットと Diameter Candidate のノードを合わせたノードセットを生成して，再帰的に Candidate Enumeration を行う． 図 2 により Diameter Candidate$< A[1], B[6] >$ において, 保留されるノードは $\{\{C[2]\},\{D[2], D[5]\}\}$ である． nested loop で $\{C[2], D[2]\},\{C[2], D[5]\}2$ 組を生成する. $Maxdistance(C[2], D[2]) < Maxdistance(A[1], B[6])$ を満たすから,$\{C[2], D[2]\}$ と DC$< A[1], B[6] >$ を合わせて, ノードセット $\{A[1], B[6], C[2], D[2]\}$ を生成する (図 4)． しかし，$\{C[2], D[5]\}$ において $Maxdistance(C[2], D[5]) < Maxdistance(A[1], B[6])$ を満たさないから，$\{C[2], D[5]\}$ を捨てる.

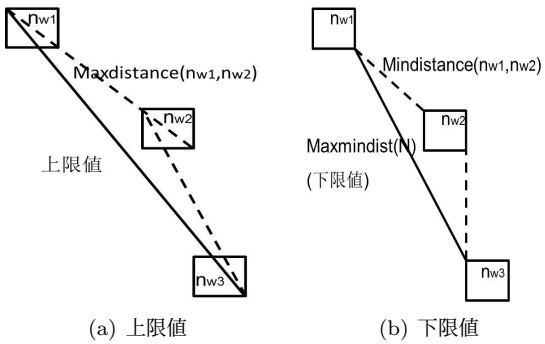
もし生成されたノードセットが全て葉ノードであれば，ノードにあるオブジェクトを抽出して同じの流れで DCC を行なって，最小なオブジェクト群を求める.

4. 枝刈り規則

2 オブジェクト $\{o_i, o_j\}$ の間の距離を $dist(o_i, o_j)$ とする． また，2 ノード $\{n_i, n_j\}$ の MBR 間の最大距離を $Maxdistance(n_i, n_j)$ とし，最小距離を $Mindistance(n_i, n_j)$ とする．次の性質が成立する:

[定理 1]:ノードセット $N = \{n_1, n_2, ..., n_m\}$ について，$\forall n_i, n_j \in N, Maxdistance(n_i, n_j)$ の最大値は N から組み合わせるオブジェクト集合の直径の上限値になっている (図 5(a)).

[定理 2]:ノードセット $N = \{n_1, n_2, ..., n_m\}$ について，$\forall n_i, n_j \in N, Mindistance(n_i, n_j)$ の最大値は N から組み合わせるオブジェクト集合の直径の下限値になっている．この値を $Maxmindist$ と表記する (図 5(b)).



(a) 上限値　　(b) 下限値

図 5: ノードセット N の上限値と下限値

DCC 方式は定理 1 のノードセットの上限値によって組み合わせの順番を決める．ノードセットの下限値によって，下記の補題 1 の性質がある:

[補題 1]:クエリ $Q = \{w_1, w_2, ..., w_m\}$ に対応したノー

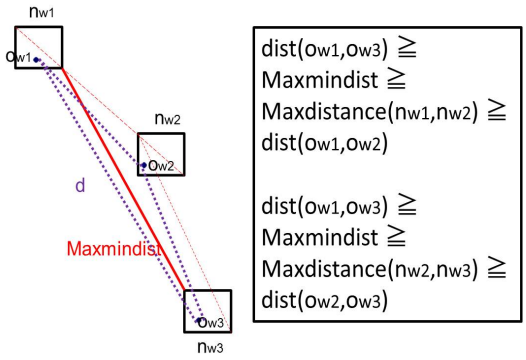


図 6: 補題 1 の例

ドセット $N = \{n_{w_1}, n_{w_2}, ..., n_{w_m}\}$ について，もしあるノード n_{w_i} と他の全てのノード $n_{w_t}(t = 1, 2, ..., m$ 且つ $t \neq i)$ の間に，常に $Maxdistance(n_{w_i}, n_{w_t}) < Maxmindist(N)$ が成立すれば，ノード n_{w_i} をノードセット N の δ^* の計算対象から外す．

例えば，図 6 において，ノードセット $N = \{n_{w_1}, n_{w_2}, n_{w_3}\}$ において，ノード n_{w_2} と n_{w_1}(また n_{w_3}) の間の $Maxdistance$ は n_{w_1} と n_{w_3} の間の $Mindistance$(すなわち下限値 $Maxmindist(N)$) より小さい．つまり，任意のオブジェクト o_{w_1} と $o_{w_3}(dist(o_{w_1}, o_{w_3}) = d)$ において，$o_{w_2}(dist(o_{w_2}, o_{w_1}) < d$ 且つ $dist(o_{w_2}, o_{w_3}) < d$ となる) が存在する．そのため，w_2 に関連するオブジェクトはオブジェクト集合の直径にならない．故に，w_1 と w_3 のオブジェクトを対象とした計算だけでも直径を求められる．

そうすると，もともと $\{n_{w_1}, n_{w_2}, n_{w_3}\}$ から展開すべきであるが，補題 1 を通して $\{n_{w_1}, n_{w_3}\}$ だけ展開してもいいので，子ノードセット或はオブジェクト集合の数を減らせる．

しかし，補題 1 はノードセット N から作る子ノードセットを組み合わせる時だけ有効であるが，N と同じ階層のノードセットの候補数を減らすためには直接有効ではない．そこで，補題 1 を用いて以下の 2 つの枝刈り規則を与える:

[枝刈り規則 1]: クエリ $Q = \{w_1, w_2, ..., w_m\}$ に対して，対応したノードセット $N = \{n_{w_1}, n_{w_2}, ..., n_{w_m}\}$ のノードを，次の 2 種類に分ける: $N = \{n_{w_a}, ..., n_{w_b}, |n_{w_c}, ..., n_{w_d}\} w_a, w_b, w_c, w_d \in Q$. ただし，$n_{w_a}$ から n_{w_b} が補題 1 によって外せないノードであり、n_{w_c} から n_{w_d} が補題 1 によって外せるノードである．この時，$\{n_{w_a}, ..., n_{w_b}\}$ を真に含むノードセットは計算する必要がない．

枝刈り規則 1 によって n_{w_a} から n_{w_b} を組合わせた時の最小直径は d である．n_{w_a} から n_{w_b} をサブノードセットとした他のノードセット $\{n_{w_a}, ..., n_{w_b}, |n'_{w_c}, ..., n'_{w_d}\}$ の直径は d より小さくないので，計算する必要がない．枝刈り規則 1 を通して，ループでノードセットを形成する途中で，もしこのようなサブノードセットが発見されれば，後の計算が必要がなくなるので，ノードセットの列挙コストを減らすことができる．

さらに DCC 方式によって，先に Diameter Candidate(DC) を作っておいて，DC から近傍のノードと連携してノードセットを生成する場合，枝刈り規則 1 の特殊形として枝刈り規則 2 を用意する:

[枝刈り規則 2]([枝刈り規則 1] の特殊形): $\{n_{w_u}, n_{w_v}\}$ を DC として生成したノードセットのうち，あるノードセット N について，n_{w_u}, n_{w_v} 以外のすべてのノードが補題 1 によって外せるなら，この DC から他のノードセットを更に生成する必要はない．

枝刈り規則 1 によって，$\{n_{w_u}, n_{w_v}\}$ をサブノードセットとしたノードセットを生成しないので，DC から他のノードセットを生成をしないようになった．そうすると，DC からノードセットを生成する場合，一旦枝刈り規則 2 のようなノードセットを発見しさえすれば，該当 DC の展開が終わるので列挙コストが減る．

5. 実験

mCK 検索では，データの分布は検索効率に大きい影響を及ぼす．そのため，本稿の実験データでは，keyword の数を 100 として，各 keyword に 500 個のレコードを関連づけ，正規分布で座標を生成する．即ちキーワード 1 つあたり，ランダムでイベント発生点を 1 つ選んで，イベント発生点との距離を正規分布で 500 個のデータを生成する．例えば，Flickr の写真データにおいて，東京タワーというタグをつけている写真データは東京タワー付近にある可能成が高い．Grid は 100 分割/level とした．

検索時，問い合わせキーワードに関するデータをロードして，On-demand で Grid を作成してから，探索を行う．

5.1.DCC についての影響

正規分布の標準偏差 σ を一辺の長さ R の 1/4 とする ($\sigma = 1/4R$)．入力 keyword の数（m） vs. 実行時間 (ミリ秒)(Java, Intel Xeon 2.67GHz, 100 回平均) を図 7 に示す．実行時間はデータのロードから，結果を出すまでの時間である．NLwP はノードの組み合わせ (ノードセット) の列挙をノード番号順のみによって行う単純入れ子ループ方法である．

実験結果からみると，DCC は NLwP 法より探索効率が優れるとわかった．10 キーワードの場合，On-demand で Grid の生成時間を含み，実行時間が 1 秒以下になる．各キーワードに関連するデータは一定程度で密集して，関連するノードの数が減少できる．また，NLwP 法は最初に良い閾値を求められないが，DCC は小さい直径の候補を優先的に選び，枝刈り能力が高い．

5.2. 枝刈り規則の効果

次に 4 節の枝刈り規則を DCC に入れた方法で評価する．ここでデータ分布について，σ は $1/4R$ と $1/8R$ の二つの場合を分ける．図 8 はこの二つの場合の 3 キーワードに関連するデータの分布を表す．図 8(b) は図 8(a) よりデータがもっと密集するので，答えの直径は図 8(b) のほうが大きい．

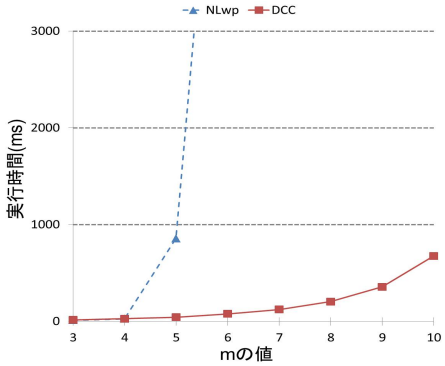
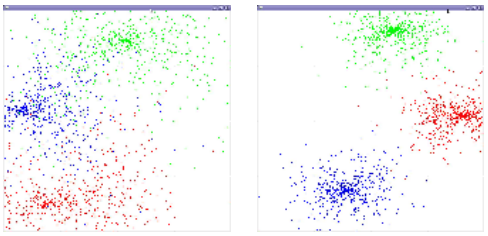


図 7: DCC の結果 ($\sigma = 1/4R$)



(a) $\sigma = 1/4R$　(b) $\sigma = 1/8R$

図 8: データの分布

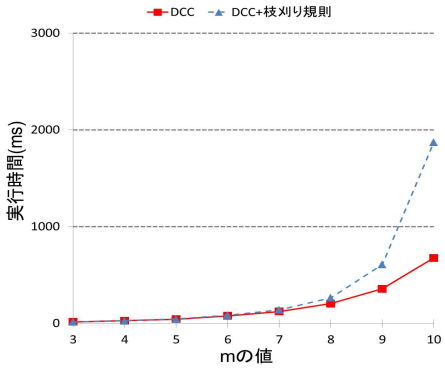
実験結果を図 9 に示す．図中,「DCC」は 3 節の DCC のみの方式で,「DCC+枝刈り規則」は 4 節の規則を追加した方式である．図 9(a) は $\sigma = 1/4R$ の場合の入力 keyword の数（m） vs. 実行時間 (ミリ秒) の結果である．結果からみると，枝刈り規則を使うと逆に探索時間が増加する．原因は Top-1 解の直径が m=10 で $0.04R$ ぐらいしかなく，ノードセットの内、に枝刈り規則の条件を満さないものが多く，その一方で，枝刈り規則の判定自体に計算コストがかかるからである．

図 9(b) は $\sigma = 1/8R$ の場合の入力 keyword の数（m） vs. 実行時間 (ミリ秒) の結果である．結果からみると，枝刈り規則が有効になる．この場合，解の直径が m=8 で $0.3R$ ぐらいである．補題 1 により多くのノードを外すことができる．特に「DCC+枝刈り規則」の方式において，枝刈り規則 2 で DC からノードセットの生成数がだいぶ減少することができるので，探索時間が単純な DCC より短くなる．そのため，全域或は局所的にキーワードごとにデータが密集する場合，上記の枝刈り規則が有効であるとわかった．

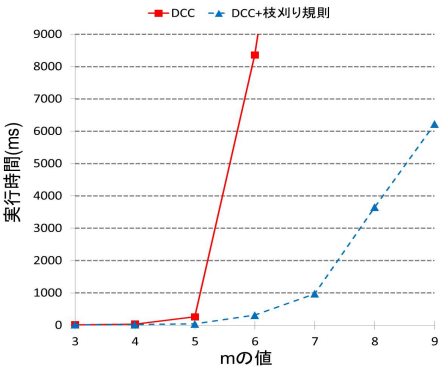
6. おわりに

本稿は mCK 問題における DCC 方式の改良方法を述べ，その性能を評価した．実行処理効率について，On-demand で Grid 構造を生成する場合，キーワードあたり 500 レコードの平面上正規分布，問合せあたり 10 キーワード程度でも，1 秒以下で計算できるとわかった．

また，ノードセットの下限値を用いて，ノードセットの列挙数を減少する枝刈り規則を設けた．実験によって，キーワードごとデータが密集している場合，枝刈り規則が有効であるとわかった．



(a) $\sigma = 1/4R$



(b) $\sigma = 1/8R$

図 9: 枝刈り規則を使う結果

参考文献

[1] D.X.Zhang et al, ”Keyword Search in Spatial Databases: Towards Searching by Document,” IEEE ICDE, pp.688-699, 2009.

[2] 邱, 他, ”空間データベースにおける m-最近接キーワード検索の一方式” 情報処理学会全国大会 2014 5M-1.