D-030

再帰的な DCC 戦略による mCK 検索の高速化
Optimization of mCK Search by using Recursive DCC Strategy

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，位置情報を持つ空間 Web データにおいて，情報抽出や検索などを扱う「空間キーワードクエリ」が注目されている．この空間キーワードクエリ問題のうち，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 個のオブジェクト群のうち最も空間的に近接している組 (最小直径を持つ) を探す問題として mCK 検索 [1] がある．

mCK 検索の効率を高めるために，探索空間において走査する対象 (オブジェクトセット) を減少することが目標である．この目標のため，いろいろな組みあわせや枝刈り方式が提案された：Zhang らは事前に全てのオブジェクトを 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，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や，複数データ集合を統合して mCK 検索を行う空間統合検索も考えられるため，我々は問合せ発生の時に On-demand で Grid を生成してから mCK 検索を行う方式として，ノードセット (或はオブジェクトセット) の列挙の優先順を制御できる探索戦略 DCC(Diameter Candidate Check) を提案している [2][3].

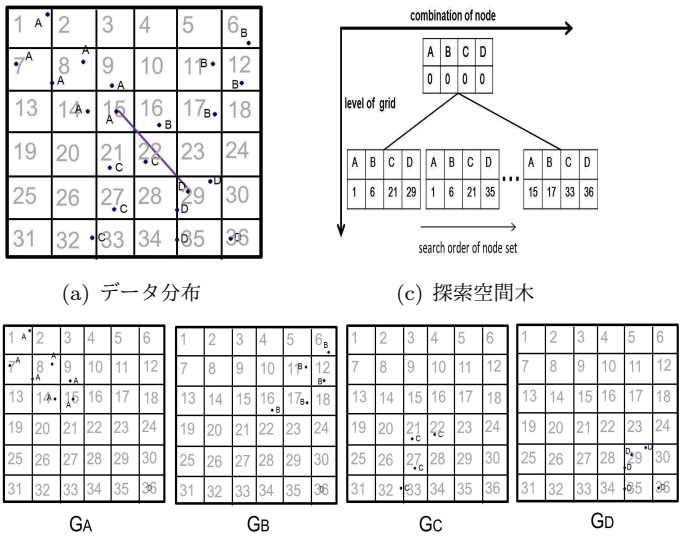
DCC 戦略は 2 ノード (或は 2 オブジェクト) ペア (Diameter Candidate，DC と略) の列挙によって優先順を決めて，各 DC からノードセット (或はオブジェクトセット) の生成のため，対象ノード (或はオブジェクト) を小さい範囲内にクラスタリングすることである．しかし，検索結果の直径が増加するとともに，クラスタの中でノードセットの生成対象の規模が大きくなるので，列挙効率が不安定になる．そのため，本稿はクラスタ内の生成対象をさらに再分割する再帰的な DCC 戦略を提案する．そしてノードセットを枝刈りする時，ノードの MBR 間の最小距離 (Mindist) を使って枝刈りする代わりに，ノード間の最小オブジェクトペアの距離 (Obj-Mindist) を使って枝刈り効率を高める方法を述べる．最後に On-demand Grid の条件で，再帰的な DCC 戦略の評価を行い，元の DCC 戦略より有効性を示す．

2.DCC の概要と問題

2.1.DCC の概要

例として，データの分布を図 1(a) に示す．クエリのキーワード $\{A, B, C, D\}$ を受けると，図 1(b) のように，On-demand により 4 つの Grid$\{G_A, G_B, G_C, G_D\}$ が生成される．図の数字はセル番号を表す (ここで 36 分割/レベル)．記号 $A[i]$ は Grid G_A の i 番目のセルであり，1 ノードと呼ぶ．そして，各 Grid から 1 ノードを選んで，合わせてサイズ m の「ノードセット」を作る．$\{A[i], B[j], C[k], D[l]\}$ は 1 ノードセットを表す．mCK 探索問題は図 1(c) のように，ノードセットを要素とした木構造になる．1 ノードセットを木の 1 要素とする (図 1(c))．

mCK 検索の効率を高めるため，探索空間木で走査するノードセットの数を減少すべきである．一般的に深さ優先探索の場合，より小さい直径を優先的に求めて閾値とすると，探索空間の枝刈り能力が高い．図 1(a) において，ノードセット $\{A[15], B[16], C[22], D[29]\}$ を先に計算すれば，最も小さい直径が探される．そのため，我々はノードセット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]．DCC は全てのサイズ m のノードセットを生成せず，サイズ 2 のノードペア (持つ直径の上限値) の列挙を通して探索順番を決める方法である．



(a) データ分布 (c) 探索空間木

(b) On-demand Grid

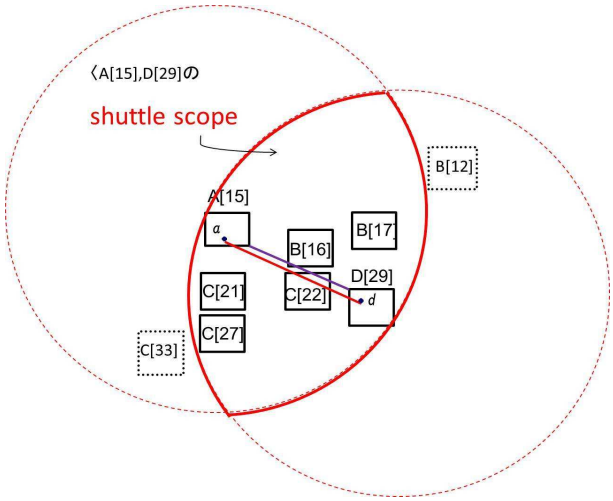
図 1: Grid と探索空間

DCC の流れについて (具体的には文献 [2] を参照)，図 1 の例により説明する：

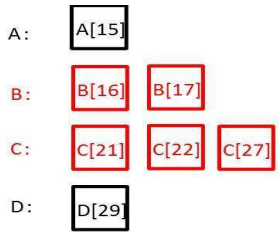
まず，Diameter Candidate(DC) の集合を作る．すなわち，全ての相異なる 2 キーワードに関連するノードのペアを Diameter Candidate とする．例えば，$\langle A[1], B[6]\rangle$, $\langle A[1], B[11]\rangle$,...,$\langle C[32], D[36]\rangle$ 計 107 を DC として作り，DC をノード MBR 間の最大距離 (Maxdist) によってソーティングをする．

次に，小さい順に DC を取り出してチェックする．該

†電気通信大学, UEC



(a) DC の shuttle sope



(b) 候補ノード

図 2: ノードセットの生成

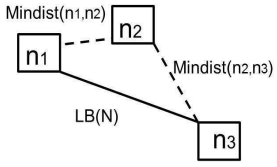
当 DC の Maxdist に関連する直径の上界値を確保するため，対象となるノードは，shuttle scope の範囲内に存在しなければならない (図 2(a) の太枠で示した)．その他のノードは生成対象外 ($B[12], C[33]$) である．もし，この範囲内に全部のキーワードが含まれないなら，該当 DC をスキップする．

最後に，shuttle scope 範囲内の候補ノードを入れ子ループでサイズ (m-2) のノードセットを組みあわせてから，DC の 2 ノードと合わせてサイズ m のノードセットを生成する．例えば，図 2(b) の B, C に関するノードを組みあわせて，DC$\langle A[15], D[29]\rangle$ に合わせると，$\{A[15], \mathbf{B[16]}, \mathbf{C[21]}, D[29]\}$,...,
$\{A[15], \mathbf{B[17]}, \mathbf{C[27]}, D[29]\}$ 計 6 ノードセットを生成する．

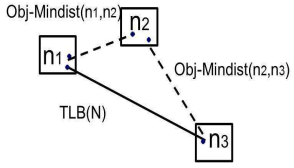
深さ優先探索のため，1 ノードセットができた後で，子ノードへ (リーフノードの場合オブジェクトへ) 展開して最小直径を求めて，この直径を閾値として他のノードセットを枝刈りする．

2.2.DCC の問題点

[問題 1]DCC は小さい DC から生成するノードセットを優先的に処理するので，より小さい直径が早めに探されることを一定程度で保証できるが，shuttle scope 内の候補ノードを単純に入れ子ループでノードセットを生成するため，同じ DC から生成するノードセットの間では，より良い探索順を保証できない．例えば，図 2 において，DC$\langle A[15], D[29]\rangle$ から生成したノードセットの内，$\{A[15], \mathbf{B[17]}, \mathbf{C[27]}, D[29]\}$ は



(a) Lower Bound



(b) Tight Lower Bound

図 3: LB と TLB

$\{A[15], \mathbf{B[16]}, \mathbf{C[22]}, D[29]\}$ を比べると，後者のほうが小さい直径を持つので優先的に生成すべきである．こうした優先順を追加したい．

[問題 2]2 ノード $\{n_i, n_j\}$ の MBR 間の最小距離を $Mindist(n_i, n_j)$ とする．ノードセット $N = \{n_1, n_2, ..., n_m\}$ において，直径の下界値を以下の式 LB で与える．

$LB(N) = max(Mindist(n_a, n_b)), \forall n_a, n_b \in N$

図 3(a) によって，ノードセット $N = \{n_1, n_2, n_3\}$ において，$LB(N)$ は n_1, n_3 の間の $Mindist$ である．

今までの枝刈り方法について，ノードセット N において，$LB(N) \geq \delta^*$(δ^*：閾値) の場合，N を枝刈りすることができる．この枝刈り方法に基づいて，最小直径を求めたものの，同じ DC から生成するノードセットを枝刈りできない問題が生じる．

図 2 では，ノードセット N_1=$\{A[15], B[16], C[22], D[29]\}$ を先に生成して $LB(N_1)$ が $A[15], D[29]$ の間の $Mindist$ であり，そこから求まる mCK 解の直径が $dist(a, d)$ である．その後，$dist(a, d)$ を閾値 δ^* としてノードセット N_2=$\{A[15], B[16], C[21], D[29]\}$ を生成する場合，$LB(N_2)$ も $A[15], D[29]$ の間の $Mindist$ である．しかし，この $Mindist$ が $dist(a, d)$ より小さいので，$LB(N_2) < \delta^*$ と判定され，N_2 の探索は枝刈りできない．つまり，$A[15], D[29]$ の $Mindist$ を LB とするノードセットは枝刈りできない．このように，もし $dist(a, d)$ が $A[15], D[29]$ のオブジェクト間の距離の最小値である場合，即ち DC $A[15], D[29]$ に関する最小直径となるので，当該 DC から他のノードセットを展開しても，$dist(a, d)$ 以下の直径を求められないのに，計算しなければならない．

再帰的な DCC と Tight Lower Bound(TLB) を使って枝刈りをする方法を提案する．

3. 再帰的な DCC と Tight lower bound

問題 1 について，同じ DC からより良い順でノードセットを生成するため，再帰的な DCC 方法を考える．再帰的な DCC の流れを説明する．

ステップ 1: 全体のノード集合を対象として全部の DC を生成し，ソートする．

ステップ 2: 小さい順に DC を選ぶ．今、DC_1 を選んだとすると，DC_1 の shuttle scope 範囲内のノードのうち，DC_1 以外のノードから更に DC を生成する．この処理を再帰的に $\lfloor m/2 \rfloor$ 回繰り返して，毎回 2 キーワードに関連するノード (DC の 2 ノード) を選ぶので，最

後に scope 範囲内が 0(m が偶数) 或は 1(m が奇数) キーワードに関連するノードしか残っていない．
ステップ 3：選んだ $\lfloor m/2 \rfloor$ 個の DC のノードと，最後の 0 或は 1 のノードを合わせてサイズ m のノードセットを生成する．

そうすると毎回小さい DC を選ぶので，ノードセットが優先順で生成される．

図 1(a) の例で，再帰的な DCC の流れを説明する．事前に $CurSet$ という変数を設定し，$CurSet$ はノードセットを生成するため既に選んだ DC を格納して用いる．$\emptyset$ と初期化する．全体のノード集合に対して，全部の DC の集合 (DCS_1) を列挙して，小さい順で DCS_1 の DC を選ぶ．$\langle A[15], D[29] \rangle$ を選ぶ場合，$CurSet$ に入れる ($CurSet = \{A[15], D[29]\}$)．そして，DC $\langle A[15], D[29] \rangle$ の scope 内に B と C に関するノードを対象として，もう一度 DC の集合 (DCS_2) を列挙して，順番で DCS_2 の DC を選ぶ．DC $\langle B[16], C[22] \rangle$ が一番小さいので，優先的に選び，$CurSet$ に入れる ($CurSet = \{A[15], D[29], B[16], C[22]\}$)．この時 $CurSet$ が全 m キーワードを含むため，$CurSet$ を展開して一番小さい直径を求める．その後，$B[16], C[22]$ を $CurSet$ から外して，DCS_2 から次の DC を選んで，$CurSet$ に入れて新しいノードセットを生成する．そうすると，DC から全部のノードセットを生成せず，$\lfloor m/2 \rfloor$ 回 DC の列挙を通して，優先順ができる．特に，ある DC の scope 内候補ノードが多いかつ m が大きい場合，生成効率が単純な DCC より高い．

再帰的な DCC の本質は scope 範囲内の対象を DCC 方式によって順番を決める方法である．

再帰的な DCC を使うと同じ DC からより小さいノードセットを先に生成するが，上記の問題 2 が依然として存在する．$\{A[15], B[16], C[22], D[29]\}$ によって求めた直径 $dist(a, d)$ を閾値とする場合，LB を通しては，他のノードセットを枝刈りできない (図 2)．そのため，以下のように示す下界 TLB を用いる (図 3(b))：

[Obj-Mindist] 2 ノード n_1, n_2 において，
$Obj\text{-}Mindist(n_1, n_2) = min(dist(o_a, o_b))$,
$(\forall o_a \in n_1, \forall o_b \in n_2.)$
2 ノードの $Obj\text{-}Mindist$ はノードの間一番小さいオブジェクトペアの距離である．

[Tight Low Bound(TLB)] ノードセット N=$\{n_1, n_2, ..., n_m\}$ において，
$TLB(N) = max(Obj\text{-}Mindist(n_a, n_b)), \forall n_a, n_b \in N$
ノードセットの TLB は任意 2 ノードの $Obj\text{-}Mindist$ の最大値である．

LB の代わりに TLB を使うと，もともと枝刈りできないノードセットを枝刈りできるようになる．図 2(a) によって，$\{A[15], B[16], C[22], D[29]\}$ によって直径 $dist(a, d)$ を閾値 δ^* とする場合，次のノードセット $N_2 = \{A[15], B[16], C[21], D[29]\}$ の TLB が $A[15], D[29]$ の間の $Obj\text{-}Mindist$(即ち $dist(a, d)$) である．そこで，$TLB(N_2) = \delta^*$ になり，つまり N_2 に関する直径が δ^* より小さくないので，枝刈りすることができる．

更に，TLB を再帰的な DCC に導入する．上記の $CurSet$ はノードセットの部分セットである．もし，$TLB(CurSet) \geq \delta^*$ の場合，$CurSet$ を部分セットとする全てのノードセットを枝刈りすることができるという性質がある．そこで，閾値を更新次第，$TLB(CurSet) \geq \delta^*$ と判定すれば，ノードセットの生成が止まる．

ノードセット N の TLB の計算コストは，N の内の 2 ノードの $Obj\text{-}Mindist$ を計算することである．この計算は Grid 構造を使って Top-Down 方式で求めるため，$m = 2$ の mCK 探索コストであり，$m > 2$ の場合に比べて高速に計算できる．

4. 実験

mCK 検索について，データの分布によって検索効率に大きい影響に及ぼすため，本稿は以下の 3 つの人工生成データ集合と Flickr データ集合を用いて実験を行う．(人工生成データは 100 キーワード，1000 レコード/キーワードとする)．

1. 均一分布：全てのレコードをランダムで生成する．
2. 正規分布 ($\sigma = \frac{1}{4}R$)：ランダムで事件点を選んで, 事件点との距離を正規分布で生成することである. 標準偏差=$\frac{1}{4}R$.
3. 正規分布 ($\sigma = \frac{1}{8}R$)：標準偏差=$\frac{1}{8}R$ の正規分布である．
4. Flickr データ：2013 年 1 月～4 月の Flickr46,303 件写真データ.

本論文は以下の 3 つのアルゴリズムを比較する：

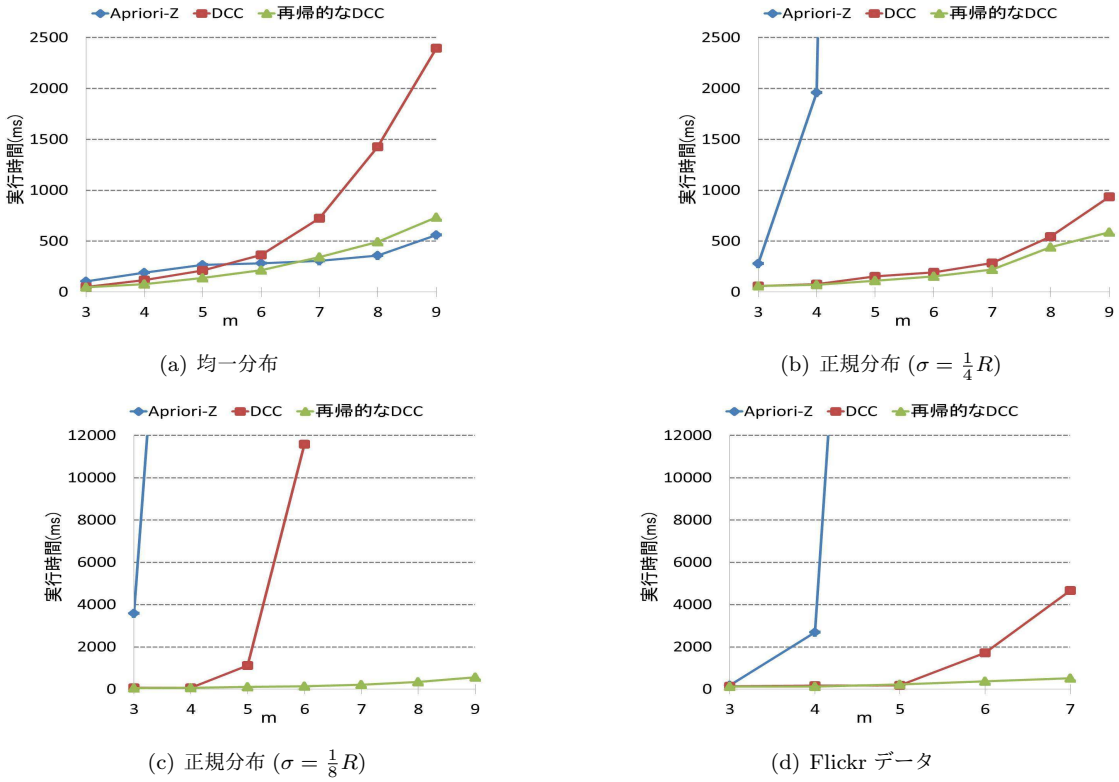
- Apriori-Z: Zhang らの Apriori 方法
- DCC：単純な DCC 方法 [2]
- 再帰的な DCC(TLB を使用)

検索の時，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 100 セル/レベルの Grid を作成してから，探索を行う．

各データ集合について，入力 keyword の数（m） vs. 実行時間を図 4 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．

図 4(a) は均一分布の場合の実験結果を表す．この分布によって，全キーワードに関するオブジェクトセットが小さい場所に集まっているため，Apriori-Z が容易に小さい直径を探されるので，実行時間が最も安定している．DCC はキーワードの増加に応じ，生成するノードセットの数が増加するので，実行時間が長くなる．隣接なノードから組みあわせるノードセットが DCC を通して枝刈りできない場合，再帰的な DCC は TLB を使って，枝刈り可能になるので，実行時間の増加が緩やかになる．

図 4(b) は $\sigma = \frac{1}{4}R$ の正規分布の場合の実験結果である．この分布によって，データが軽く偏るが，小さい直径が存在しているとわかる．Apriori-Z が優先的に小さい直径を求められないので，効率がひどくなる．DCC も再帰的な DCC も小さい結果を優先的に探せるため，効率が良い．



(a) 均一分布

(b) 正規分布 ($\sigma = \frac{1}{4}R$)

(c) 正規分布 ($\sigma = \frac{1}{8}R$)

(d) Flickr データ

図 4: 実験結果

図 4(c) は $\sigma = \frac{1}{8}R$ の正規分布の場合の実験結果である．この分布によって，データが極めて偏るので，小さい直径が存在していない可能性がある．そこで DCC について，DC の scope 内候補ノードが多くなるから，大量なノードセットを生成するので実行時間が長くなる．再帰的な DCC がこのようなノードセットの生成を避けるため，探索時間が短い．

図 4(d) は Flickr データの場合の実験結果である．実際データの分布について，分布が把握できなくて，小さい直径が存在するかどうかわからない．Apriori-Z と DCC の効率が不安定になると判った．これに比べ，再帰的な DCC が一貫してより良い順でノードセットを生成し，また TLB を使ってノードセットの生成を減少するため，安定な効率を得られる．

5. おわりに

本稿は mCK 問題における再帰的な DCC に基づいた新しい高速化方法を提案し，その性能を評価した．従来の単純な DCC 方法と比べて，結果の直径が大きい場合でも，ノードセットの生成優先順も決められる．また，TLB を使う枝刈り方法，大量のノードセットの生成を避けられて，探索効率を高める．実験結果からみると，提案した再帰的な DCC は異なるデータ分布において，データ分布により探索効率に影響を控えることができるとわかった．

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.

[3] 邱, 他, ”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” FIT 2014 D-005.