データベースシステム 116－38
（１９９８．７．９）

高速検索のための時空間データ管理の一方式
～動的データへの対応～

才脇直樹，仲篤起，野沢博，西田正吾
大阪大学大学院　基礎工学研究科

近年，データベースの分野において，時間と空間の情報を共に扱う時空間データベースの研究が盛んにおこなわれている．しかし，時間・空間データを効率良く扱うデータ管理手法はいまだに確立されていないのが現状である．我々は従来，複数状態を有するデータの管理を対象とするAdaptive Tree構造を提案してきたが，本稿では特に動的に変化するデータを取り扱う手法に焦点を当てて論じている．

まず，従来のAdaptive Tree構造について概説し，その有効性をST・3D管理構造との計算機によるシミュレーション結果の比較・検討によって示した．次に，動的データを管理するための拡張手法を提案して，計算機シミュレーションにより評価した．その結果，動的データに対してもAdaptive Tree構造で効率的に管理可能であることが示された．

Spatio-Temporal Data Management for Efficient Search

Naoki SAIWAKI, Atsuki NAKA, Hiroshi Nozawa and Shogo NISHIDA
Graduate School of Engineering Science, OSAKA University

This paper deals with data management structures for efficient search for large scale spatio-temporal data. We propose Adaptive Tree Structure (We call the AT Structure), in which either spatio data structure or temporal data structure is selected adaptively depending on the demand of search. This paper especially describes the extension of the proposed data structure for dynamic data cases. The concrete data structure and its performance by computer simulation are discussed.

1.はじめに

最近，3次元空間データや時空間データに関する研究が活発化し，例えば時間の重要性[1]や空間データの効率的な管理手法[2]，医用データにおける時間情報の管理[3]等に注目した研究が行われている．しかし，ここで扱っているデータは主として文字・数値属性情報であり，図形等のデータに対する空間検索の効率化はあまり考慮されていない．さらに，R-treeをベースにして直角四面体を近似形状として用いるシステム[4]やマルチメディアオブジェクトの時間的関連記述手法[5]，画像オブジェクトの版管理[6]等の研究も進められている．

我々も，従来より木構造に基づいた高速検索のためのデータ構造の研究を進めてきており[7][8][9]，地下配管の設備管理への適用検討等も行ってきている[10]．本稿では，まず，筆者らが提案しているデータ構造（以下，AT構造と呼ぶ）の概要について述べる．AT構造は，時空間データに対して時間木と空間木を用意しておき，検索範囲に応じて適応的に切り替えることにより高速な検索を実現する方式であり，シミュレーションによりその有効性が確認されている．

また，時空間データを管理するデータ構造を考える際に重要な点の一つとして，対象のデータ集合が「静的」であるか「動的」であるかが挙げられる．データ構造が動的データを管理可能である

と，最新の入力データを管理できる点やデータ構造更新の際のコストを削減することができる等のメリットが生じる．本稿では特に，動的なデータを管理可能とするためのAT構造の拡張について具体的な検討を行いシミュレーション評価の結果を示す．

2.AT構造の概要[8]

多次元データの管理構造として，静止データを管理対象としたk-d木[11]，動的データに拡張したBD木[12]，MD木[13]等が提案されている．これらを応用して時空間情報を持つデータを管理する場合，代表的な手法としては，

（1）バージョンごとに木構造を作成・保持するMT構造

（2）全データを位置情報により分割した1つの木構造で管理し，時間情報はデータの属性として扱うST構造

（3）時間情報も位置情報と同一視し，位置（2次元）＋時間（1次元）の3次元空間内の時区間データとして1つの木構造で管理する3D管理構造

などが考えられる．しかし，それぞれのデータ構造には一長一短があり，ST構造のように空間木を中心にデータ構造を作成し，時間情報をデータ属性として取り扱うと，空間検索範囲が狭い場合には，効率が良くなるのに対して，空間検索範囲が広くなると効率が悪化する．一方，MT構造のように時間中心でデータ構造を作成すると，時間検索範囲が狭い場合には効率が良いのに対して，時間検索範囲が広くなると効率が悪くなる．この「検索範囲が狭い場合には検索効率が非常に良いが，検索効率が広くなるにつれて検索効率が悪化する」という性質は，木構造を用いる限り，基本的には避けられないものである．

そこで，AT構造では空間木と時間木を用意しておいて，空間と時間の検索範囲に応じて，どちらの木構造を優先して検索するかを適応的に切り換えている．具体的には，任意の時空間検索範囲に対し，「時間情報と位置情報の検索範囲の全体に対する割合を求め，小さい方を採用する」というロジックで時間木を用いるか空間木を用いるかを決め，他の条件については木構造より選択されたものの中から全数チェックを行う．

さらに，本方式を適用するにあたり時間木の検索効率を上げるために，時区間データ（線分データ）を，開始時間・終了時間の2次元平面上にマッピングして点データに変換し，3分木として管理する手法も採用した．なお，各状態に対応するために，それぞれ1つずつの時間木を用意している．

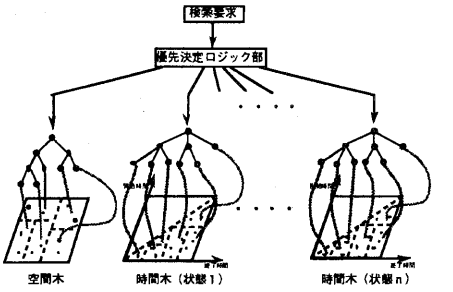


図1　拡張したAT構造の概略

つまり図1に示すように，1つの空間木と状態数に応じたn個の時間木を用意しておき，検索範囲に応じて検索優先決定ロジック部でどの時間木，もしくは空間木を検索するかを決め，他の条件については木構造より選択されたものの中から全数チェックを行う方式を採用する．

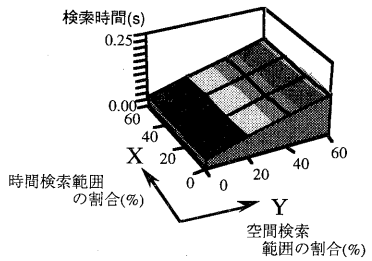
このAT構造の有効性を確かめるために，計算機によるシミュレーション実験を行った．以下にその一例を示す．

具体的には，対象データとして，位置の情報，時間の情報と状態の情報を持ったデータを考える．そのうち，位置の情報としては実数の一様乱数を発生させて，X座標，Y座標を与える．次に，時間の情報に関しては，各状態にそれぞれ開始時間と終了時間を与える．開始時間及び存在期間には整数の一様乱数を与えた．また，終了時間に関しては，存在期間を開始時間に加えることにより与えた．性能評価は，上記のデータを対象に計算機で行う．データ点数は100000点で，各葉ノードは最大10個のデータを持つことが出来るとして実験を行った．また，計算機はSiliconGraphic社のIRIX6.0 INDYを用いた．

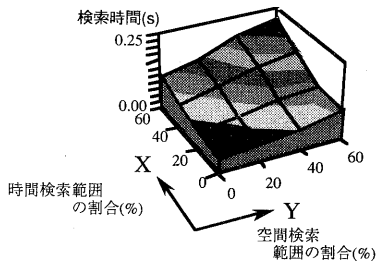
実験結果は，まず，データ構造作成時間に関しては，状態数が増えるにつれて作成する木の数が多くなるため，当然の事ながらデータ構造作成時間はほぼ状態数に比例して増加していく．次に，検索実行時間に関しては，検索のパラメータが状態毎の時間情報と空間情報という形で多数存在するため，状態数とあるパラメータを変化させた時の検索実行時間を示すことにする．

図2にST構造，3D管理構造，提案するデータ構造に対して，時間と空間の検索範囲をX，Y軸にと

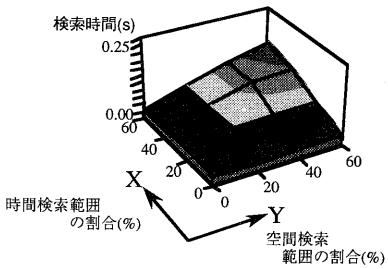
り，検索時間をZ軸にとった3次元グラフを示す．これより全検索範囲にわたり提案するデータ構造がST構造，3D管理構造より優れているというデータが得られている．



(a) ST構造の検索時間



(b) 3D管理構造の検索時間



(c) AT構造の検索時間

図2 各データ構造の検索実行時間

また，検索優先決定ロジックにかかる時間については，シミュレーションの結果，例えば，空間検索範囲が5%（各状態の検索範囲は10%に固定）の場合で約0.1msecであり，全体の検索時間約10msecに対して，ほぼ無視できる時間であることが確認できた．

3.動的データへの拡張

静的データとはデータがあらかじめすべて与えられているものであり，動的データとは逐次データが投入・削除されていくものである．静的データを対象としているデータ構造では，データ構造更新の際には全ての構造を再構築する必要が生じる．しかし動的データを管理可能なデータ構造では，投入データに応じてデータ構造の1部を再構成することが可能である．従って，動的データを管理可能なデータ構造ではデータ更新のコストを大幅に削減することができる．また他にも最新の入力データを管理できる等のメリットが生じる．

従来，動的データを扱うデータ構造[12][13]の多くは，平衡木の考え方を利用してデータ投入に応じてデータ構造を変化させるものであった．本研究では，管理対象の位置は変化せず，時間的に状態が変化するような時空間データを対象としている．このため，空間に関しては，全ての対象がある領域内に存在するという仮定をしていた．動的データを考慮した場合にも，空間木に関しては上記の平衡木の考え方を用いることができる．しかし，時間木の場合は，管理領域が更新の度に時間軸方向に拡張するという問題が生じる．これは，管理領域が閉じているという木構造を作成する際の前提条件に反している．このため，閉じた管理領域内で各ノードの領域を変化させ平衡性を保証するという従来の手法は適用できない．そこで，ここでは動的データに対応できるようにAT構造に変更を加える．以下，具体的なデータ構造の構成について述べる．

3.1 検索木決定ロジック部

AT構造では，静的データを対象とした検索木決定ロジック部において時間木と空間木について検索範囲の全体に対する割合の小さい方を検索する方式を採用していた．これは，「検索範囲が狭い場合には検索効率が非常に良いが，検索効率が広くなるにつれて検索効率が悪化する」という木構造の性質のためである．動的データを対象とする場合は，時間木の管理領域がデータ更新毎に拡張するという点を考慮せねばならないが，基本的には同様のロジックを用いる．管理領域が増加するという問題に関しては，最新データの時間領域を変数に代入し，各入力毎に管理領域の面積を逐次増減させることにより対応する．

3.2 空間木のデータ構造

空間木について，静的データを対象とした場合はk-d木を用いてデータ構造を作成していたが，k-d木を用いた動的データの管理では，データ投入毎に

データ構造の平衡性が損なわれ検索効率が悪化することが知られている[14]．そこで，ここでは平衡木であるMD木を用いることにより空間木を作成することにする．具体的には，データが投入された際にリーフのデータ容量がある個数以内になるようデータ構造を逐次変更，あるいは再構成する．このデータ容量は事前に決めておく．検索の場合はノードの持つ値と検索範囲を比較して，どちらの部分木を検索する必要があるかを判断する．この処理を再帰的におこない，リーフにある必要データを取り出す．

3.3 時間木のデータ構造

本稿では，管理対象の位置は変化せず，時間的に状態が変化するような時空間データを対象としている．このため，位置に関しては全ての対象がある領域内に存在すると仮定する．また，時間に関しても静的データを対象とした場合，ある時間までのデータを扱うといった仮定から，閉じた領域内で全ての点を管理可能にする．これはデータ構造化に木構造を用いており，管理領域が増加する場合，トップダウン的に領域を分割する木構造では増加したデータを含めてどのように再構築化するかという問題が不可避であるためである．

このように動的なデータを考慮する際に最も問題になるのは，時間情報の管理領域が増加することである．そこで，この増加領域を効率よく管理する新しい手法が必要となる．その管理にはデータの入力がおこなわれる毎に，以下の手順に従って一時的に増加部分を管理する2種類の分割法（child分割法，k-d分割法）を提案する．

（1）child 分割手法

child 分割手法では，一時的に管理領域の増加部分を管理する．この手法では，新規データが投入されるごとに次の処理によりデータを格納する．

S1. 時間情報の管理領域の情報（各ノードが所持）を最新情報へと拡張する．すなわち図3において，一番右端の分割軸を終了時間軸と平行に延長することにより増加領域を管理する．

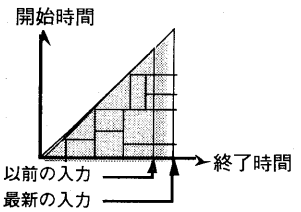


図3　時間データ構造の拡張方法

S2. 入力データを格納すべきリーフを探索する．

S3. リーフに格納されているデータ数がリーフのデータ容量より小さい場合はリーフにデータを追加して終了．データ容量と格納データ数が等しければ領域分割をおこなう．領域分割は，終了時間の軸に対して分割軸が垂直になるようにおこなう．分割後，新しいリーフを作成し，ポインタを親のノードに持たせる．（図4）

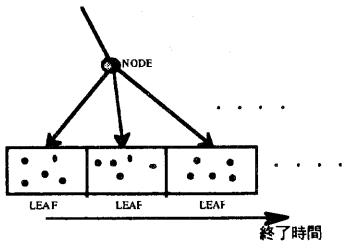


図4　リーフの分割方法（child分割法）

（2）k-d 分割手法

k-d 分割手法では，新規データ投入時に次の手順に従って一時的に管理領域の増加部分を管理する．

S1. 時間情報の管理領域の情報（各ノードが所持）を最新情報へと拡張する．

S2. 入力データを格納すべきリーフを探索する．

S3. リーフに格納されているデータ数がリーフのデータ容量より小さい場合はリーフにデータを追加して終了．データ容量と格納データ数が等しければ領域分割をおこなう．領域分割は，k-d木の分割方法を利用して，（X→Y→X→・・・→X）のように巡回的に分割軸を選択し，リーフの管理領域を2等分することによりおこなう（図5）．

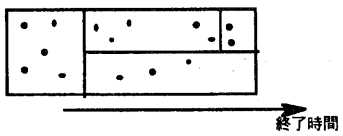


図5　リーフの分割方法（k-d分割法）

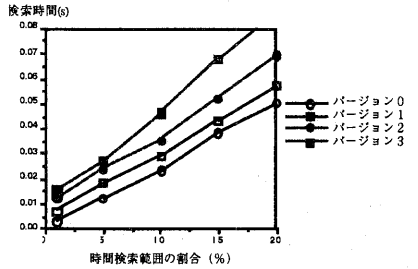
4.シミュレーション実験

ここでは，計算機実験により，提案する2手法の性能の比較を示す．実験では，まずバージョン0として100,000個，以降バージョンごとに20,000個の投入をバージョン3までおこない，各バージョンごとの時間検索性能を評価した．対象データとし

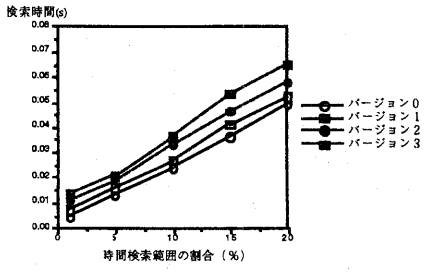
て，空間情報と時間情報を持った点情報を考える．そのうち，空間情報としては２種類の乱数を発生させ，X座標，Y座標として与える．時間情報に関しては，開始・終了時間を与える．開始時間及び存在期間には 一様乱数を与えた．また，終了時間に関しては，存在期間を開始時間に加えることにより与えた．なお，リーフのデータ容量は10，計算機はSGIのINDYを用いた．

実験では，対象データに対して作成したそれぞれの管理構造を用いて，時間検索の性能を評価した．検索の種類としては，ある期間にある範囲に存在するものを探すという範囲検索を用いた．なお，性能の評価は，範囲検索における検索時間とした．

図６に実験結果を示す．図６は空間検索範囲を一定（10%）にした場合の検索時間の変化である．この図より，バージョン数が大きくなるにつれて検索効率が悪化していることが分かる．これは，データ投入数が多くなればデータ構造の平衡性が崩れることが原因になっていると思われる．この結果より，一定数のデータが投入された時点でデータ構造を再構築して平衡性を改良する必要が生じることが分かる．



(a)　k-d 分割手法を用いた時間検索結果



(b)　child 分割手法を用いた時間検索結果

図６　動的データに対する時間検索実行時間の変化

次に，k-d 分割手法とchild 分割手法の性能の評価をおこなった．図７は時間検索範囲が２０%の場合の検索時間と時間データのバージョンとの関係を示している．この図より，時間の検索範囲が小さい場合にはk-d 分割手法もchild 分割手法も検索効率に大きな差はないが，時間の検索範囲が大きくなるにつれ，k-d 分割手法の検索効率がchild 分割手法より悪化することが分かる．これはk-d 分割手法の方がchild 分割手法よりも作成する中間ノード数が多くなるという点に原因があると思われる．

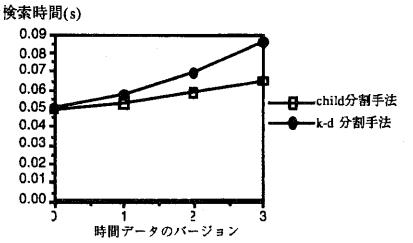


図７　child分割法及びk-d分割法の検索効率の比較

以上の実験結果より，AT構造では時間木の再構築法としてchild 分割法を採用する．

次に，AT構造の動的データに対する評価実験をおこなう．空間木はMD木で作成した．また，時間木の再構築手法としてはchild分割法を用いている．実験環境としては，SGIのINDY上でC言語を用いてプログラミングをおこなった．対象として，まずバージョン０として100,000個，以降バージョンごとに20,000個の投入をバージョン３までおこない，各バージョン毎の検索実行時間を計測する．この実験によりAT構造における動的データの管理効率を調べることができる．なお，リーフのデータ容量は空間木・時間木ともに10としている．

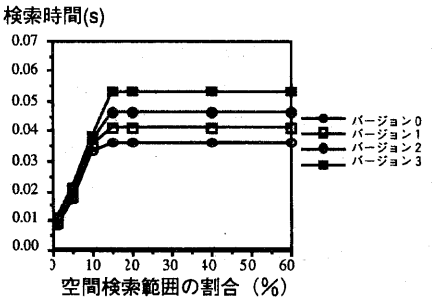


図８　動的データに対する検索効率

図８に実験結果の一例を示す．図中，横軸は全体の管理領域に対する検索範囲の割合（%）を示している．ここでは，時間検索範囲の大きさを固定（10%）にした際の検索実行時間の変化を示している．空間検索範囲の割合が10%を越えて15%以上になると検索実行時間の増加はなくなる．これは，検索木決定ロジック部により検索木が空間木から時間木へと切り替えがおこなわれたからである．

次に，バージョン変化，すなわちデータ投入数に伴う検索効率の変化について検討をおこなう．図からは検索範囲が10%以上になり時間木を検索するようになると，バージョン増加に伴い検索効率が悪化することが分かる．原因として，時間木ではデータ投入数が多くなればデータ構造の平衡性が崩れることが挙げられる．これは一定数のデータが投入された時点でデータ構造を再構築して平衡性を改良する必要が生じることを意味している．

5.まとめ

本稿では，AT構造で動的データの取扱いを可能にする方法について検討した．従来のAT構造は，「データの状態数が増加すると若干検索効率が悪化するが，ほぼ無視できる範囲である」という特性を持つことが，２節の実験結果の一例にも示されている．しかし，これは静的データを対象としていたため，最新のデータを管理することが出来なかった．

そこで，３節以下で示したような手法で動的データへの拡張を試みた．動的データの管理に関してはk-d分割とchild分割という二つの手法を提案し，性能の評価をおこなった．実験より，時間の検索範囲が大きくなるにつれ，k-d分割の検索効率がchild分割より悪化することが分かったので，child分割手法を用いて動的データの管理を試みた．

その結果，管理自体は可能であるが，データ投入数の増加に伴う時間検索効率の悪化は避けられないので，ある程度を越えるとデータ構造を再構築する必要がある，という結論が得られた．

以上，今回は対象データとして，設備や建物のように時間情報は変化するが空間情報は変化しないデータのみを扱っている．今後は，位置や形状が時間的に変化する対象に対してのデータ構造作成の可能性についても検討していく予定である．

なお，本研究は一部文部省科研費重点領域研究（高度データベース）により行われたものである．

[参考文献]

[1]増永良文「マルチメディアデータベースと時間」情報処理，情報処理最前線，Vol.36, No.5, pp.369-377, 1995.

[2]坂内正夫，大沢　裕，「画像データベース」昭晃堂，1987.

[3]A.U.Tansel, J.C.Clifford, S.Gadia, S.Jajodia, A.Segev, and R.Snodgrass, Temporal Databases : Theory, Design, and Impremention, The Benjamin/ Cummings Publishing Campany, Inc., 1993.

[4]黒木　進，牧之内顕文，「単体複体の概念を用いた時空間データモデルUniverseの設計」情報処理学会データベースシステム研究会資料，July，1996.

[5]増永良文，清水英成，「マルチメディアオブジェクト間の時間的関連記述の一フレームワーク」電子情報通信学会論文誌D-II，Vol.J79-D-II，No.4，マルチメディア論文特集，pp.492-501，April 1996.

[6]川島　亨，田幡　勝，金森吉成，増永良文，「画像オブジェクトの再利用のための版管理」情報処理学会，Proceedings of Advanced Databese System Symposium'95，pp.87-96，Dec.1995.

[7]寺岡照彦，丸山稔，中村泰明，西田正吾「空間検索を効率化した時空間データ管理構造の提案－多次元Persistent Tree－」電子情報通信学会論文誌D-II，Vol.J78-D-II,No.9, pp.1346-1355，1995.

[8]仲　篤起，才脇直樹，辻本浩章，西田正吾「時空間データの高速検索のためのデータ管理方式」，Proceedings of Advanced Database Symposium '96，pp.71-78，Dec.1996.

[9]Atsuki NAKA, Naoki Saiwaki and Shogo Nishida, "Spatio-Temporal DataManagement for Highly Interactive Environment", Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics, pp.571-576, Orland, Oct.1997.

[10] 玉田隆史，寺岡照彦，丸山稔，西田正吾　「3次元仮想都市空間を用いた設備管理システム」電気学会論文誌C,Vol. 116-C, No.5, pp.517-523, 1996.

[11] J.L.Bentley,"Multidimensional Binary Search Trees Used for Associative Searching," Commun.ACM, vol.18,pp.509-517,1975.

[12] 大沢　裕，坂内正夫，「良好な動特性を持つ多次元点データ管理構造の一提案」電子通信学会論文誌D, vol.J66-D, No.10, pp.1193-1200, Oct.1983.

[13] 中村泰明，阿部　茂，大沢　裕，坂内正夫，「多次元データの平衡木による管理 - MD木の提案」電子通信学会論文誌D，vol.J71-D，No.9，pp.1745-1752，Sept.1988.

[14] 中村泰明，阿部　茂，大沢　裕，坂内正夫，「空間的広がりを持つ図形データのMD木による管理」，電子情報通信学会論文誌D-II，Vol.J73-D-II，No.12，pp.1976-1984，1990.