Explorando a Multidimensionalidade da *Kd-Tree* para Suporte a Temporalidade em Dados Espaciais Vetoriais do Tipo Ponto

LEONARDO RODRIGUEZ HEREDIA[1]
CIRANO IOCHPE[1]
JOÃO COMBA[1]

[1]Instituto de Informática da UFRGS. Av. Bento Gonçalves, 9500, bloco 4. Agronomia. 91.501-970.Porto Alegre, RS.
{leonardo, ciochpe, comba}@inf.ufrgs.br

Abstract. The representation of time in geographic information systems (GIS) has been subject of intense investigation in late years. Actual research involves from modeling techniques for spatio-temporal representation of reality to spatio-temporal data structures and file organizations. In this paper, we propose to explore the multidimensional structure of both the *Kd-Tree* and the *Adaptive Kd-Tree* in order to keep the validation time or transaction time of the points stored in them.

1. Introdução

Sistemas de Informação Geográfica (SIG) são sistemas computacionais capazes de capturar, armazenar, consultar, manipular, analisar e imprimir dados referenciados espacialmente à superfície da terra [11]. Esses dados, que possuem atributos relacionados a um sistema de coordenadas, são denominados dados georreferenciados ou simplesmente dados geográficos.

Um SIG se caracteriza por permitir ao usuário realizar operações complexas de análises sobre dados geográficos, envolvendo seus aspectos espaciais e descritivos. O Banco de Dados Geográficos (BDG) é o componente do SIG responsável por estruturar e armazenar os dados geográficos de forma a possibilitar a realização das operações de análise [10].

Para facilitar o acesso aos aspectos ou atributos espaciais da informação geográfica, o BDG utiliza estruturas de dados adequadas para o armazenamento, indexação e manipulação de feições espaciais. Dentre essas estruturas, destacam-se as *Quadtree*s [15], as *Kd-Trees* [15] e as *RTrees* [8].

As estruturas de dados espaciais dividem-se basicamente em dois tipos: matriciais e vetoriais. Nas estruturas matriciais, a região contendo a informação espacial é constituída por uma matriz, ou grade de células, onde, para cada célula dessa matriz, existe um valor associado. Nas estruturas vetoriais, cada informação espacial caracteriza um objeto (feição) que pode ser representado por diferentes tipos de geometria como pontos, linhas e polígonos.

A representação dos aspectos espaço-temporais da realidade geográfica ainda não é suportada pela maioria dos SIG comerciais hoje existentes. Entretanto, diversos trabalhos de pesquisa estão sendo realizados no sentido de incorporar a dimensão temporal aos SIG, tanto no nível conceitual [9,4,6,7,14] como no nível físico ou de armazenamento da informação [1,2,3,8,12,13,15,17,18].

A temporalidade da realidade geográfica pode ser representada, tanto nos atributos descritivos, como nos espaciais. Um atributo espaço-temporal de um dado geográfico é constituído por uma informação espacial (geometria georreferenciada) associada a um rótulo temporal. Cada rótulo temporal associa à feição um valor de tempo de transação, de tempo de validade, ou mesmo de ambos (sistema bitemporal).

Para o armazenamento e indexação dos objetos espaço-temporais, a maioria dos autores propõe extensões a estruturas de dados espaciais já existentes. Tais estruturas são adaptadas para que possam armazenar o rótulo temporal associado ao dado espacial.

No caso de estruturas espaciais baseadas em árvores, a sobreposição de árvores é uma das alternativas utilizadas. A técnica de sobreposição de árvores consiste em criar uma árvore para cada versão dos dados ao longo do tempo, sendo que os caminhos não alterados, ao longo do tempo, são compartilhados pelas árvores. A *Overlapping Linear Quadtree* [17], a *Multiversion Linear Quadtree* [18] e a *Temporal RTree* [19,20] são exemplos de estruturas espaço-temporais baseadas na sobreposição de árvores.

Neste artigo, explora-se a característica multidimensional das estruturas do tipo *Kd-Tree* e *Adaptive Kd-Tree* [15] para que estas possam suportar tempo de validade ou tempo de transação em dados espaciais vetoriais do tipo ponto. Diferentemente de outras propostas, esta não se baseia na sobreposição de árvores para a incorporação do rótulo temporal, mas na característica

multidimensional nativa das estruturas *Kd-Ttree* e *Adaptive Kd-Tree*.

Como parte significativa da realidade geográfica é representada por pontos em aplicações atuais de SIG, e pelo fato de as estruturas do tipo *Kd-Tree* serem eficientes para consultas de proximidade entre pontos, acredita-se que a proposta deste artigo, inclusive por sua simplicidade, possa contribuir para o projeto e a implementação de SIG espaço-temporais.

O restante do artigo está organizado da seguinte forma. A seção 2 recapitula as estruturas originais da *Kd-Tree* e da *Adaptive Kd-Tree*. Na seção 3 são propostas as estruturas espaço-temporais *TKd-Tree* e *Adaptive TKd-Tree*. Na seção 4, discute-se os resultados de testes comparativos entre a *TKd-Tree* e a *TR-Tree* [19,20]. Finalmente, a seção 5 apresenta as conclusões e possíveis trabalhos futuros.

2. Revisando as Estruturas de Dados do Tipo Kd-Tree

A *Kd-Tree* [15] é uma árvore de busca binária de *k* dimensões que se caracteriza por testar, em cada nodo percorrido, um valor chave. Esse valor chave está associado a um *discriminador*. O *discriminador* indica qual dimensão da *Kd-Tree* foi utilizada para determinar o valor chave que divide os nodos restantes em duas sub-árvores. Os nodos com os valores dessa dimensão maiores que o valor chave são armazenados do lado direito e os restantes do lado esquerdo do nodo percorrido. A *Kd-Tree* é eficiente para consultas que envolvam proximidade. As consultas de proximidade geralmente são feitas com base em um ponto central e um raio de busca. O resultado é um conjunto de pontos contidos dentro dessa região circular (área de buffer).

Há, basicamente, dois tipos de *Kd-Trees*: a *Kd-Tree* original, que será referenciada tão somente como *Kd-Tree Original* e a *Adaptive Kd-Tree*. A diferença entre ambas está na política de escolha do *discriminador*, bem como no próprio armazenamento do dado espacial nos nodos da árvore.

2.1 Kd-Tree Original

Na *Kd-Tree Original* o valor do *discriminador* é o valor de uma das *k* dimensões de cada nodo da árvore. A escolha da dimensão que será utilizada como *discriminador* é realizada de forma alternada. Por exemplo, para uma *Kd-Tree Original* de duas dimensões, denotadas por *x* e *y*, o valor do *discriminador* do primeiro nível é o valor da dimensão *x*. O próximo *discriminador* é o valor de *y*. No próximo nível, o *discriminador* volta a ser o *x* e assim sucessivamente.

Cada nodo de uma *Kd-Tree* Original armazena o próprio dado. No caso de feições vetoriais do tipo ponto, a *Kd-Tree Original* armazena em cada nodo as dimensões dos pontos. Para pontos bidimensionais, as dimensões armazenadas são as coordenadas x e y.

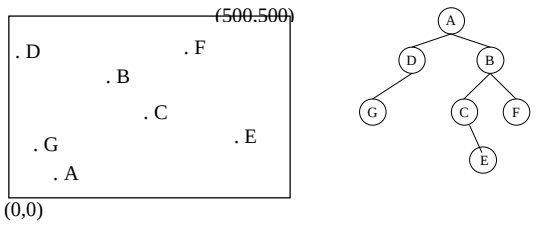
Cada nodo de uma *Kd-Tree Original* contém no mínimo *k + 4* campos. No caso de uma *Kd-Tree Original* para pontos no plano, cada nodo possui um total de seis campos. Desses seis campos, dois são utilizados para armazenar as coordenadas x e y do ponto; outros dois são os ponteiros para a sub-árvore da esquerda e para a sub-árvore da direita; um campo é o identificador do nodo; e o último campo é utilizado para identificar a coordenada (*x* ou *y*), utilizada como *discriminador* nesse nível.

O processo de inclusão de pontos em uma *Kd-Tree Original* é relativamente simples e semelhante ao processo de inclusão de dados em uma árvore binária. Para cada novo ponto a ser inserido na árvore, percorre-se a árvore verificando em cada nodo percorrido a posição em que esse novo ponto será inserido. Essa verificação da posição ocorre a partir da comparação do valor de uma das coordenadas do novo ponto com uma das coordenadas do nodo da árvore. A coordenada escolhida para a comparação é indicada pelo *discriminador* do nodo da árvore que está sendo percorrido. Se o valor da coordenada do novo ponto for menor ou igual à coordenada do nodo da árvore, então se percorre a sub-árvore à esquerda do nodo da árvore. Caso contrário, o caminhamento continua a partir da sub-árvore da direita. Caso o ponteiro para a sub-árvore a ser percorrida contiver o valor nulo, então o novo ponto é inserido nesse local e o ponteiro do nodo da árvore é atualizado, passando a apontar para o novo ponto.

A figura 2 apresenta um exemplo de uma *Kd-Tree Original* de duas dimensões construída para armazenar os pontos contidos na tabela 1. Os pontos foram inseridos na árvore na ordem em que aparecem na tabela.

Tabela 1 – Pontos a serem armazenados

Id	**X**	**Y**
A	100	114
B	133	306
C	182	194
D	34	384
E	448	168
F	341	385
G	79	150



(a) (b)

Figura 2 – (a) Região 500x500 contendo os pontos da tabela 1. (b) *Kd-Tree Original* resultante.

A disposição dos nodos na *Kd-Tree Original* depende diretamente da ordem em que os pontos foram inseridos na árvore. Essa característica pode resultar em uma árvore com uma distribuição desproporcional dos nodos das sub-árvores.

2.2 Adaptive Kd-Tree

A *Adaptive Kd-Tree* [15] constitui uma alternativa de otimização da *Kd-Tree Original*. Diferentemente da *Kd-Tree Original*, na *Adaptive Kd-Tree* os pontos ficam armazenados apenas nos nodos folhas. Os nodos intermediários armazenam um valor associado a um *discriminador*, que são as informações utilizadas para a construção e para o caminhamento correto na árvore.

Outra diferença entre a *Adaptive Kd-Tree* e a *Kd-Tree Original* está na política de determinação do *discriminador* e do valor associado ao mesmo. No caso da *Kd-Tree Original* o *discriminador* é escolhido alternando as dimensões dos pontos e o valor associado ao *discriminador* é o valor da própria coordenada da dimensão. Entretanto, na *Adaptive Kd-Tree* é possível adotar outras alternativas para a determinação da dimensão a ser utilizada como *discriminador* e do seu valor. O ideal é verificar qual a dimensão, e qual o valor associado a essa dimensão, que melhor divide os pontos em dois conjuntos. Para obter esse resultado podem ser utilizados métodos estatísticos como média, mediana e variância.

No item 1 da seção 2 foi verificado que na construção da *Kd-Tree Original* os pontos são inseridos na ordem em que aparecem. Essa ordem de inclusão dos pontos influi diretamente na estrutura final da árvore.

Entretanto, na *Adaptive Kd-Tree*, a árvore é construída a partir de um conjunto de pontos já existentes. Ou seja, para a construção é necessário que se tenha todos os pontos que serão armazenados.

Devido a essa característica de construir a árvore utilizando funções que melhor dividam o conjunto de pontos em duas partes, a *Adaptive Kd-Tree* acaba gerando uma estrutura com os nodos distribuídos de forma mais organizada que a *Kd-Tree Original*. O resultado dessa melhor distribuição dos nodos na estrutura é a possibilidade de uma melhor performance na realização das consultas de proximidade, pois há grandes chances de que menos nodos sejam percorridos no caminhamento.

A figura 3 apresenta a *Adaptive Kd-Tree* construída a partir do conjunto de pontos da tabela 1.

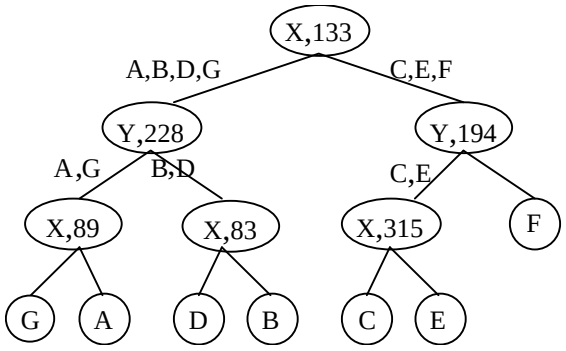


Figura 3 – *Adaptive Kd-Tree*

2.3 Consulta na *Kd-Tree Original* e na *Adaptive Kd-Tree*

O processo de consulta na *Kd-Tree Original* e na *Adaptive Kd-Tree* é praticamente o mesmo. Conforme mencionado, no início da seção 2, as *Kd-Trees*, em geral, são eficientes em consultas que envolvam buscas por proximidades. Esse tipo de consulta se caracteriza por receber, como parâmetros, um ponto central e um raio de busca. O resultado obtido é um conjunto de pontos que estão contidos nessa área de busca.

A forma de percorrer a estrutura da árvore é a mesma, tanto para a *Kd-Tree Original*, como para a *Adaptive Kd-Tree*. Visando facilitar o entendimento desse processo, o exemplo a seguir descreve os passos básicos necessários para um consulta de proximidade em uma *Kd-Tree* de duas dimensões.

Exemplo de consulta:

Consultar todos os pontos contidos em uma região de centro C(*x*,*y*) e raio de *d* unidades.

Algoritmo da consulta:

- determinar o menor e maior ponto da região de busca. O menor ponto será o ponto *CMenor(x-d,y-d)*, e o maior ponto será o ponto *CMaior(x+d,y+d)*, onde *x* e *y* são as coordenadas do ponto central da região de busca e *d* é o raio de busca fornecido.
- após a determinação do menor e maior ponto da área de busca, inicia-se o caminhamento na árvore a partir do nodo raiz. Para cada nodo percorrido, deve ser feita a seguinte análise. Comparar o valor associado ao *discriminador* com uma das

coordenadas do menor ponto – *CMenor* – da área de busca. A coordenada a ser compara é aquela indicada pelo *discriminador*. Se a coordenada comparada de *CMenor* for maior que o valor associado ao *discriminador*, então não é necessário percorrer a sub-árvore à esquerda do nodo atual. Caso contrário deve ser feita a comparação entre uma das coordenada de *CMaior* com o valor associado ao *discriminador*. A coordenada de *CMaior* a ser comparada também é a indicada pelo *discriminador*. A partir dessa comparação, se a coordenada comparada de *CMaior* for menor que o valor associado ao *discriminador*, então não é necessário percorrer a sub-árvore à direita do nodo analisado.

- esse processo deverá ser feito recursivamente até que se atinjam os nodos folhas da árvore.
- no caso da consulta em *Kd-Tree Original*, como cada nodo armazena as próprias coordenadas dos pontos, uma função deverá ser realizada em cada nodo percorrido para verificar se esse ponto está dentro da região de busca.
- na consulta em *Adaptive Kd-Tree*, a função para verificar se o ponto do nodo está dentro da área de busca só deverá ser realizada nos nodos folhas, uma vez que, como foi visto no item 2 da seção 2, a *Adaptive Kd-Tree* armazena as coordenadas dos pontos apenas nos nodos folhas.
- o resultado dessa consulta é um conjunto de pontos que estavam dentro da região de busca.

Como pode ser observado, no processo de busca em *Kd-Trees*, seja a *Kd-Tree Original* ou a *Adaptive Kd-Tree*, existe grandes chances de se percorrerem apenas alguns nodos da árvore para se encontrar os pontos que satisfazem a condição de consulta. Essa possibilidade de poupar processamento, analisando apenas alguns nodos é que caracteriza as *Kd-Trees* como estruturas de índices para dados multidimensionais, ou seja, dados descritos por duas ou mais coordenadas.

3. Incorporando Temporalidade à *Kd-Tree* a partir de sua Multidimensionalidade

A *Kd-Tree* é, por natureza, uma estrutura de dados para armazenamento e indexação de dados multidimensionais. Sendo assim, essa estrutura tem capacidade de armazenar e indexar dados com duas, três ou mais dimensões.

Em SIG, um atributo espacial vetorial do tipo ponto é, geralmente, caracterizado por ter duas dimensões. Essas dimensões são as coordenadas geográficas, usualmente representadas por *x* e *y*, representando a longitude e a latitude do ponto.

O tempo pode ser percebido como uma, ou duas dimensões adicionais ao dado espacial. Se for considerado o tempo de transação, esse poderia ser representado por apenas uma dimensão adicional. Essa dimensão representaria o instante de tempo em que esse ponto foi inserido no BDG. Entretanto, para que seja possível manter um histórico das remoções dos pontos, seria necessária mais uma dimensão além da dimensão contendo o instante de inclusão, que guardaria o instante de tempo em que esse ponto foi removido do BDG.

No caso de se utilizar o tempo de validade, esse seria representando por duas novas dimensões na estrutura que armazena o ponto. Dessa forma, além das dimensões espaciais *x* e *y*, cada ponto possuiria mais duas dimensões associadas a ele. Essas dimensões seriam temporais, que armazenariam dois instantes de tempo, representando o intervalo em que o ponto é válido no sistema.

A idéia proposta nessa seção é justamente acrescentar duas novas dimensões aos pontos armazenados nas *Kd-Trees*. Essas duas novas dimensões podem ser de tempo de transação ou tempo de validade. Através dessa incorporação das duas novas dimensões temporais, torna-se possível o armazenamento de pontos temporais, bem como a realização de consultas espaço-temporais sobre esses pontos.

Alguns aspectos temporais foram considerados na proposta, bem como nos exemplos apresentados:

- o tipo de tempo considerado para tempo inicial e tempo final pode ser o tempo de validade ou o tempo de transação;
- o eixo temporal [5] é linear e totalmente ordenado;
- o rótulo temporal *now* indica o tempo atual do sistema;
- o tempo é discreto, podendo variar de 0 até *now* (para tempo de transação);
- o *chronon*[5] considerado é uma unidade de tempo representada pelo número inteiro 1.

A figura 4 apresenta um exemplo de um nodo de uma *Kd-Tree* com as dimensões temporais incorporadas.

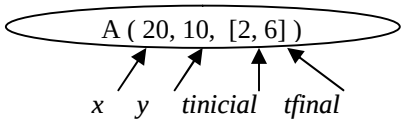


Figura 4 – Nodo temporal de uma *Kd-Tree*

Se na figura 4 for considerado o tempo de transação, as dimensões *tinicial* e *tfinal* representam o instante em que o ponto foi inserido e excluído da estrutura, respectivamente. Para tempo de validade, *tinicial* e *tfinal* determinam o intervalo em que o ponto é válido.

3.1 Kd-Tree Original Temporal (TKd-Tree)

A *Kd-Tree Original Temporal*, ou simplesmente *TKd-Tree*, é uma estrutura de dados baseada na *Kd-Tree Original* [15] que incorpora a dimensão temporal aos pontos armazenados nessa estrutura.

Na *Kd-Tree Original*, as coordenadas espaciais dos pontos ficam armazenadas nos nodos da árvore. A estrutura da *TKd-Tree* é basicamente igual a da *Kd-Tree Original*. A diferença está na inclusão de um rótulo temporal em cada nodo da árvore. Esse rótulo temporal, conforme foi visto no início dessa seção, é formado por dois instantes de tempo, podendo ser tempo de validade ou tempo de transação. Sendo assim, uma *TKd-Tree* para armazenamento de pontos no plano possui efetivamente quatro dimensões. As duas dimensões cartesianas, denotadas por *x* e *y*, e as duas dimensões temporais, denotadas por *tinicial* e *tfinal*.

O processo de construção da *TKd-Tree* é o mesmo utilizado na geração da *Kd-Tree Original*. As duas novas dimensões temporais são tratadas da mesma maneira que as dimensões espaciais. Os *discriminadores* são escolhidos alternadamente em cada nível, iniciando pela dimensão *x*, seguida da dimensão *y*, da dimensão *tinicial* e da dimensão *tfinal*. Após um nível com *discriminador tfinal*, o próximo *discriminador* será novamente a dimensão *x* e assim sucessivamente.

Tabela 2 – Pontos de exemplos paras as *Kd-Trees* temporais

Id	x	Y	tinicial	tfinal
A	100	114	2	4
B	133	306	1	1
C	182	194	6	8
D	34	384	0	2
E	448	168	1	3
F	341	385	3	3
G	79	150	2	2

A figura 5 apresenta um exemplo de uma *TKd-Tree* construída a partir dos pontos presentes na tabela 2.

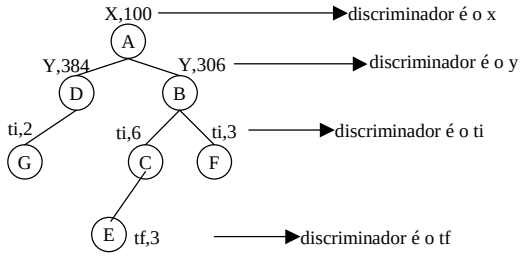


Figura 5 – *TKd-Tree*

3.2 Adaptive Kd-Tree Temporal (Adaptive TKd-Tree)

A *Adaptive Kd-Tree Temporal*, ou simplesmente *Adaptive TKd-Tree*, constitui um estrutura baseada na *Adaptive Kd-Tree* que incorpora mais duas dimensões, que são as dimensões temporais de tempo de validade ou tempo de transação inicial e final.

A inclusão das dimensões temporais na *Adaptive TKd-Tree* segue os mesmos princípios adotados para a incorporação temporal na *Kd-Tree Original*, abordado no item anterior.

A diferença entre a *Adaptive TKd-Tree* e a *TKd-Tree* está justamente na característica que diferencia essas duas estruturas nas suas versões originais não temporais. Ou seja, na *TKd-Tree*, as dados dos pontos ficam armazenados tanto nos nodos intermediários como nos nodos folhas. Na *Adaptive TKd-Tree*, apenas os nodos folhas armazenam os valores associados aos pontos. Os nodos intermediários armazenam o *discriminador* e o valor associado a esse *discriminador*.

O processo para a determinação da dimensão que servirá de *discriminador* bem como o valor associado ao *discriminador* é o mesmo adotado na *Adaptive Kd-Tree* não temporal, mencionado no item 2 da seção 2 desse artigo.

Para a construção da *Adaptive TKd-Tree* é necessário que se tenha todos os pontos que serão inseridos na estrutura. O procedimento de geração da árvore é o mesmo utilizado para a construção da *Adpatative Kd-Tree*.

Visando simplificar o processo de construção da *Adaptive TKd-Tree*, serão utilizadas nesse trabalho as seguintes políticas para a determinação da dimensão que servirá de *discriminador*, bem como do valor associado a esse *discriminador*:

- a dimensão do *discriminador* será escolhida alternadamente em cada nível, iniciando pela dimensão *x*, seguido da dimensão *y*, da dimensão *tinicio* e finalmente seguida da dimensão *tfinal*. No próximo nível após o nível com *discriminador* em

tfinal, o processo deverá ser reiniciado, começando novamente pela dimensão *x*.

- o valor associado ao *discriminador* de cada nível será a média aritmética dos valores da coordenada indicada por esse *discriminador*.

Seguindo as regras citadas acima e aplicando os passos para construção da *Adaptive TKd-Tree* no conjunto de pontos disponíveis na tabela 2, obtém-se como resultado a estrutura apresentada na figura 6.

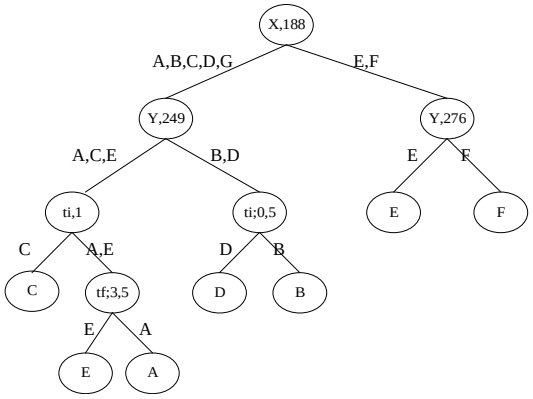


Figura 6 – *Adaptive TKd-Tree*

3.3 Consultando Kd-Trees Temporais

Utilizando as *Kd-Trees Temporais* torna-se viável a realização de consultas espaço-temporais envolvendo pontos. As consultas de proximidade, por exemplo, passariam a ter mais dois parâmetros, que seriam o tempo inicial e o tempo final dos pontos.

O processo de consulta nas *Kd-Trees Temporais*, tanto na *TKd-Tree* como na *Adaptive TKd-Tree*, se difere das versões não temporais dessas estruturas apenas no momento em que o nodo percorrido contém como *discriminador* uma dimensão temporal.

Para verificação de qual sub-árvore deverá ser percorrida em um nodo cujo *discriminador* é uma dimensão temporal o seguinte critério deverá ser aplicado:

- Caso o *discriminador* do nodo seja a dimensão *tinicial*:
    - Se o tempo final passado por parâmetro pela consulta for menor ou igual ao valor associado ao *discriminador* do nodo analisado, então apenas a sub-árvore da esquerda deverá ser percorrida;
    - Senão as duas sub-árvores deverão ser percorridas.
- Caso o *discriminador* do nodo seja a dimensão *tfinal*:
    - Se o tempo inicial passado por parâmetro pela consulta for maior ou igual ao valor associado ao *discriminador* do nodo analisado, então apenas a sub-árvore da direita deverá ser percorrida;
    - Senão as duas sub-árvores deverão ser percorridas.

Para um melhor entendimento do processo de busca espaço-temporal em uma *Kd-Tree Temporal*, o exemplo abaixo mostra a realização passo a passo de uma consulta de proximidade espaço-temporal na *TKd-Tree* ilustrada na figura 5.

Exemplo de consulta por tempo de validade:

Retornar todos os pontos que estão dentro de uma região circular com centro em P(440,157), raio de 30 unidades e válidos no intervalo de tempo de validade [3;3].

Resolução da consulta:

- Passo 1. Determinar *PMenor* e *PMaior*

  O resultado é *PMenor*(410,127) e *PMaior* (470,187)

- Passo 2. Iniciar o caminhamento na árvore

  Comparar o valor da coordenada *x* de *PMenor* com a coordenada *x* do nodo raiz. Como 410 é maior que 100, então constata-se que apenas a sub-árvore da direita deverá ser percorrida.

- Passo 3. Análise do nodo *B*

  Agora o *discriminador* é a coordenada *y*. Comparar a coordenada *y* de *PMaior* com a coordenada *y* do nodo *B*. Verifica-se que 187 é menor que 306. Nesse caso apenas a sub-árvore da esquerda deverá ser percorrida.

- Passo 4 . Análise do nodo *C*

  O *discriminador* desse nível é a dimensão *tinicial*. Deve-se então comparar o valor do tempo de validade final passado por parâmetro pela consulta com o valor associado ao *discriminador* desse nodo, nesse caso o próprio valor da dimensão *tinicio* do nodo atual. Como 3 é menor que 6, então apenas a sub-árvore da esquerda deverá ser percorrida.

- Passo 5. Análise do nodo *E*

  Como o nodo *E* é uma folha, deve-se apenas realizar o cálculo para verificar se ele está espacialmente dentro da área de busca e também verificar se o ponto é válido no intervalo de tempo de validade fornecido como parâmetro da consulta.

É importante ressaltar que, para cada nodo percorrido, deve ser feito um teste para verificar se o mesmo está dentro ou fora da área de busca. Deve ser verificado também se ele está dentro, ou fora, do intervalo de tempo de validade, fornecido como parâmetro da consulta.

4. Testes comparativos

Para analisar a eficiência da estrutura apresentada, foram realizados testes comparativos entre a *TKd-Tree Original* e uma outra estrutura proposta na literatura. A estrutura escolhida para comparação foi a *Temporal Rtree (TRTree)* [19,20]. A escolha da *TRTree* se justifica por ser uma estrutura que pode ser utilizada para indexação de pontos com variação de tempo de transação e por apresentar vantagens em termos de desempenho se comparada com outros métodos [1].

Os algoritmos dos dois métodos foram implementados utilizando um ambiente de programação visual. Os testes comparativos foram executados em um mesmo computador com processador de 1.67GHz e memória RAM de 512Mb. O tamanho de página utilizado na geração da *TRTree* foi de 512 bytes.

Os testes utilizaram conjunto de dados com 10000, 20000, 50000, 75000 e 100000 pontos. Esses pontos foram gerados de forma aleatória e suas coordenadas estavam contidas em uma janela de (0,0) a (1000,1000). Também foram gerados cinco conjuntos de consultas espaço-temporais por intervalo de tempo. Esses conjuntos de consultas continham 100, 500, 1000, 1500 e 2000 consultas espaço-temporais simples, do tipo: ***Selecionar todos os pontos contidos dentro da região retangular definida por (xi,yi,xf,yf) e presentes no instante de tempo t.***

Em cada consulta desses cinco conjuntos, as coordenadas da região retangular, bem como o instante de tempo *t*, eram diferentes e foram gerados de forma aleatória.

Foram executados três tipos de testes. O primeiro teste comparou o tamanho dos arquivos de índice gerados pelas duas estruturas. Os resultados obtidos estão na Tabela 3. O segundo teste verificou o tempo de construção do índice. Os valores resultantes estão na Tabela 4. Nesses dois testes foram utilizados os cinco conjuntos de pontos descritos anteriormente, cada um com 10000, 20000, 50000, 75000 e 100000 pontos. O terceiro teste comparou o tempo de resposta das duas estruturas para um conjunto de consultas espaço-temporais por instante de tempo. O conjunto de pontos utilizados para construção dos índices nesse último teste foi o de 100000 pontos. Foram realizadas 100, 500, 1000, 1500 e 2000 consultas espaço-temporais em cada índice contendo os 100000 pontos. Os resultados obtidos estão na Tabela 5.

Tabela 3 – Tamanho do índice

Conjunto de Pontos	Tamanho (kb)	
	TKd-Tree	TRTree
10000	312	1568
20000	625	3147
50000	1563	7899
75000	2344	11813
100000	3125	15811

Tabela 4 – Tempo de construção do índice

Conjunto de Pontos	Tempo(s)	
	TKd-Tree	TRTree
10000	0,3	2
20000	1	5
50000	5	14
75000	8	22
100000	12	30

Tabela 5 – Tempo de resposta para consultas espaço-temporais

Quantidade de consultas	Tempo(s)	
	TKd-Tree	TRTree
100	0,03	0,5
500	0,3	1,5
1000	1	8
1500	2,8	13
2000	3,1	20

5. Conclusões e Trabalhos Futuros

A principal conclusão obtida, com esse trabalho, é que as *Kd-Trees* temporais propostas, tanto a *TKd-Tree* como a *Adaptive TKd-Tree* possibilitam o armazenamento correto e a posterior consulta a objetos espaciais vetoriais do tipo ponto com a dimensão temporal associada.

Observando os resultados dos testes realizados, percebe-se que a *TKd-Tree* apresenta vantagens no que se refere a uso do espaço em disco, tempo de construção e tempo de resposta para consultas por instante de tempo, se comparada com a *TRTree*.

Como já foi mencionado, a *TKd-Tree* é uma estrutura de armazenamento e acesso a pontos. Por esse motivo, não foram realizados testes com o uso da *TKd-Tree* para dados do tipo linha ou polígono. Para esses tipos de geometrias, de acordo com estudos realizados[1], a *TRTree* aparece como uma das melhores opções.

Seguem, como sugestões para trabalhos futuros, os itens descritos a seguir:

- testes de desempenho com outros tipos de consultas espaço-temporais como, por exemplo, consultas por intervalo de tempo;
- comparações entre o desempenho de consultas espaço-temporais na *TKd-Tree* e *Adaptive TKd-Tree* com outras estruturas de índices espaço-temporais que suportem o tempo de validade.

Referências

[1] Almeida, Victor T. Realização Eficiente de Consultas em Bancos de Dados EspaçoTemporais. Rio de Janeiro: COPPE/UFRJ, 2001.

[2] Becker, B., Gschwind, S., Ohler, T., et al., 1996, "An Asymptotically Optimal Multiversion B-tree", *The VLDB Journal*, v. 5, n. 4 (Dec), pp. 264-275.

[3] Bliujute, R., Jensen, C. S., Saltenis, S., et al., 1998, "R-tree Based Indexing of Now-Relative Bitemporal Data", In: *Proceedings of 24rd International Conference on Very Large Data Bases*, pp. 345-356, New York City, New York, August..

[4] Botelho, M.A. Incorporação de facilidades espaço-temporais em bancos de dados orientados a objetos. São Paulo: IMECC-UNICAMP, 1998.

[5] Edelweiss, N. Banco de Dados Temporais: teoria e prática. In: Congresso Nacional da SBC, 18., Jornada de Atualização em Informática, 17., Belo Horizonte, MG, 1998. Anais... Belo Horizonte, MG: UFMG, 1998.

[6] Faria, G. Um banco de dados espaço-temporal para desenvolvimento de aplicações de sistemas de informação geográfica**.** Campinas: Instituto de Computação da UNICAMP, 1996.

[7] Goralwalla, I. A.; Ozmu, M. T.; Szafron, D. An Object-Oriented Framework for Temporal Data Models. In: Etzion, O.: Jajodia, S.; Sripada, S. Temporal Databases: research and practice. Berlin: Springer-Verlag, 1998, p.367-405. (Lecture Notes in Computer Science, v.1399)

[8] Guttman, A. R-Trees: A Dynamic Index Structure for Spatial Searching. In Proceedings of ACM SIGMOD Conference on Management of Data, pages 599-609, June 1984.

[9] Langran, G. Time in Geographic Information Systems. London: Taylor & Francis, 1992.

[10] Lisboa, Jugurta; Iochpe, Cirano. Specifying analysis patterns for geographic databases on the basis of a conceptual framework. In: ACM symposium on advances in geographic information systems, 7., 1999, Kansas City, USA. Proceedings... Kansas City: ACM Press, 1999.

[11] Maguire, D.J.; Goodchild, M.F.; Rhind,D. (Eds.) Geographical Information Systems; Principes and Applications. 2 Vol., Longman Scientific & Technical, 1991.

[12] Manolopoulos, Y., Kapetanakis, G., 1990, "Overlapping B+-trees for Temporal Data", In: *Proceedings of the 5th Jerusalem Conference on Information Technology (JCIT)*, pp. 491-498.

[13] Mota, J**.** Estendendo *Quadtrees* para Suporte ao Armazenamento e Recuperação de Dados Espaço-Temporais. Porto Alegre: CPGCC da UFRGS, 2000.

[14] Rocha, L.V.; Edelweiss, N. *GeoFrame-T: A Temporal Conceptual Framework for Data Modeling* Proceedings of the th ACM International Symposium on Advances in Geographical Information Systems - ACM-GIS 2001, Atlanta, GA, USA, November 9-10, 2001. p.47-52

[15] Samet, H. The desing and analysis of spatial data structures. 2 ed. Addison Wesley, 1990.

[16] Saltenis, S., Jensen, C., Leutenegger, S., et al., 2000, "Indexing the position of continuously moving points", In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pp. 331-342, Dallas, Texas, May.

[17] Tzouramanis, T. Overlapping linear *quadtrees*: a spatio-temporal access method. In: ACM-GIS, 1998. Proceedings…, Bethesda, MD, November 1998, p. 1-7.

[18] Tzouramanis, T. Multiversion Linear Quadtree for Spatio-Temporal Data. In: ADBIS-DASFAA'2000, 2000. Proceedings…, Prague, Czech Republic, September 2000.

[19] Zimbrão, G., Almeida, V.T., Souza, J.M. "The Temporal R-Tree", Technical Report ES492/99, COPPE/Federal University of Rio de Janeiro, Brazil, March 1999.

[20] Zimbrão, G.; Souza, J.M.; Almeira, V.T. "Efficient Processing of Spatiotemporal Queries in Temporal Geographical Information Systems". In: 6th International Conference on Information Systems, Analysis and Synthesis ISAS 2000. Orlando, Florida, USA, July 2000.