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

2006－CVIM－153（13）
2006／3／16

事例ベース対象追跡・認識のための近さ優先探索グラフの対話的構築アルゴリズム

坂平 星弘† 和田 俊和† 坂垣内 洵也† 加藤 丈和†

†和歌山大学 システム工学部 情報通信システム学科
〒640-8510 和歌山県 和歌山市 栄谷 930

E-mail : †{s_saka,twada,sakajun,tkato}@vrl.sys.wakayama-u.ac.jp

あらまし 我々は，対象の様々な見えを蓄積した画像データベースを用いた対象追跡・認識法を提案している．このアルゴリズムでは，近さ優先探索グラフ (NFTG) と呼ばれるデータ構造を用いた擬似的な最近傍探索が行われるが，この NFTG を構築するためには，画像データ間の距離を全て計算しておく必要があるため，データ数を N，画像の次元数を d とした場合，$O(dN^2)$ の計算が必要となり，対話的に対象のデータを追加していく用途には適さない．本報告では，対象の見えを反映した NFTG を即座に計算するアルゴリズムを提案する．

キーワード 近さ優先探索 NFTG 対象追跡と認識 逐次グラフ構成

An incremental NFTG construction algorithm for interactive target tracking and recognition

Seihiro Sakahira†, Toshikazu Wada†, Junya Sakagaito†, Takekazu Kato†

†Department of Computer and Communication Science, Wakayama University,
930 Sakaedani, Wakayama, 640-8510, Japan

E-mail : †{s_saka,twada,sakajun,tkato}@vrl.sys.wakayama-u.ac.jp

Abstract We have proposed an image retrieval based object tracking and recognition method. In this method, nearest neighbor (NN) search within an image database is performed for the object image extracted from input frame. For this search, we have proposed an approximate NN search algorithm based on Nearest First Traversing Graph (NFTG), which utilizes the previous search result as the starting point of the next search. However, the computational complexity of this graph construction is $O(dN^2)$, where N and d are the number of images in the database and the dimensionality of each image data respectively. This construction algorithm is not suitable for the interactive application that stores user specified object images one by one. In this paper, we propose an incremental tree construction algorithm whose complexity is $O(dN)$ for storing a single object image. Through the experiments, we confirmed the effectiveness of our new graph construction algorithm.

Keywords Nearest First Traversing, NFTG, Object Tracking and Recognition, Incremental graph construction

1 序論

我々は，画像データベースを用いた対象追跡・認識の手法として，Nearest First Traversing Graph（NFTG）というグラフ構造を用いた擬似最近傍探索法を提案し，数千枚の対象画像を用いて約 250fps という速度で追跡認識を行うことに成功している [1]．NFTG の実用性はパターン空間内でそれぞれのパターンに対して近いパターンと辺を結ぶことにより，擬似的ではあるが，高速な最近傍探索を行なうことが出来るところにある．

従来の NFTG 構成アルゴリズムでは予め全ての

パターン (学習データ) を与えて記憶しておき，全パターン間の距離を計算し，構成するという手法が用いられている．この方法では構成時間 (主に距離計算時間) が非常にかかってしまい，次元数 d，頂点数 N の場合構成コストは $O(dN^2)$ になり，次々に追加されていくパターンに対して，効率的なグラフ構築は行えない．これは，追跡・認識対象の画像をユーザが与えることができる対話的システムを構築する際の大きな障害となる．

そこで本論文では NFTG に次々に追加されていくパターン一つに対して NFTG 部分的に変化させ構成するという逐次的な構成アルゴリズムを提案する．最初に提案する方法は入力と既存の頂点間の距離を全て計算し，部分的にグラフを変化させて構成していく方法である．この方法は現在最も高速なアルゴリズムとなっており，1 パターン追加によるグラフ構成時間を 0.015s(次元数 1000 で 1500 点目を入力した場合の構成時間) という結果が得られている．この場合の構成コストは次元数 d，頂点数 N の場合，$O(dN)$ である．この逐次的に構成した NFTG のことを incremental Nearest First Traversing Graph(iNFTG) と呼ぶことにする．

上述した手法を用いても，構成コストは次元数と，頂点数が増えれば増える程かかってしまう．そこで，最もコストがかかる距離計算時間を減らす手法として，本論文では辺分割手法を提案する (3.4 小節で述べる)．この方法は入力と全ての頂点の距離を計算するのではなく，辺の方向を用いる．具体的にはそれぞれの辺の垂直二等分線を計算し，その線で頂点を分けて構成していくという方法である．

以下，2 章ではまず NFTG が満たすべき性質について述べ，3 章では iNFTG の構成方法について述べる．そして 4 章では評価実験を行なった．

2 近さ優先探索グラフ:NFTG

本章では従来手法である NFTG の性質について説明する．NFTG が持つべき性質は以下の三つである．

連結性： すべての頂点は，連結でなければならない．

単調性： 任意の頂点からスタートして NstFT で頂点を移動させるたびに，ターゲットと，現在位置との距離は，単調に減少しなければならない．

極小性： グラフのどの一辺でも取り除くと上記の二つの性質が満足されなくなる．すなわち，冗長性のないグラフでなければならない．

これらの性質のうち，単調性が満足されれば，連結性は満足される．なぜならば，連結でなければ単調性は満足されないからである．また，単調性を満たさないグラフにある辺を追加することによって単調性（と連結性）が満足されたとすると，少なくとも冗長なグラフにはなっていないと言える．グラフ構築時に追加する辺がある頂点に関する単調性を成立させるために必要不可欠な辺であれば，このようにして出来上がるグラフは極小性も満足すると言える．

この 3 つの性質は第 3 章以降で述べる iNFTG でも満たさなければならない性質であるが，少なくとも極小性についてはそれほど重要ではない．多少冗長であったとしても，実際のグラフとして十分機能するからである．iNFTG では極小性は満たされていないが，十分にグラフとして用いることが出来ることを確認している．

3 逐次的近さ優先探索グラフ:iNFTG

本章では NstFT によって最近傍探索が可能なグラフの NFTG を逐次的に構成する方法について述べる．

iNFTG 構成方法の一つめの手法として入力と既存の頂点間の全ての距離を最初に計算しておく手法を提案する．この手法を 3.1 節で述べ，本論文ではこの手法を全距離計算手法と呼ぶことにする．

二つめの手法としてそれぞれの辺を垂直二等分線で分割し，各頂点が垂直二等分線で分けられた空間のどちらに存在するかという情報だけを用いて iNFTG を作成する方法について提案する．この方法を辺分割手法と呼び，3.4 節で述べる．

3.1 iNFTG 構成アルゴリズム 1 - 全距離計算手法

全距離計算手法について述べる．全距離計算というのは，入力 n と既に iNFTG を構成している頂点群 (既存頂点群と呼ぶことにする) との距離を全て計算し，必要だと思われる部分のみを変化させる手法である．

このアルゴリズムの流れは大きく分けて三つのステップに分けられ, 頂点が一点追加されるごとに以下のステップを辿る．

1. 距離計算
2. 辺追加
3. 辺削除

距離計算については NFTG の構成方法と同様である．辺の追加は連結性を満たすために存在する．

辺追加で追加した辺は，別の入力が来た場合に不用な辺となる場合が多数出現する (極小性を満たさない辺)．それらの辺を削除しなければ iNFTG 全体のグラフサイズが巨大になってしまう．グラフサイズが巨大になるということは構成コストが大きくなり，また，NstFT を行なう場合，各頂点の平均分岐数 (頂点が持っている辺数の平均) が大き

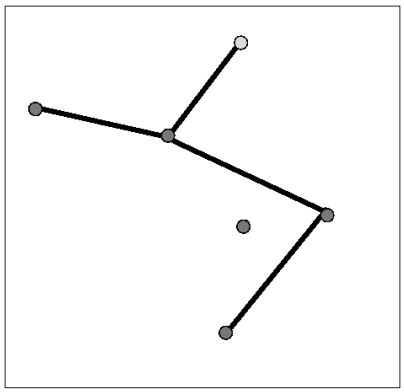

図 1: 頂点入力直後

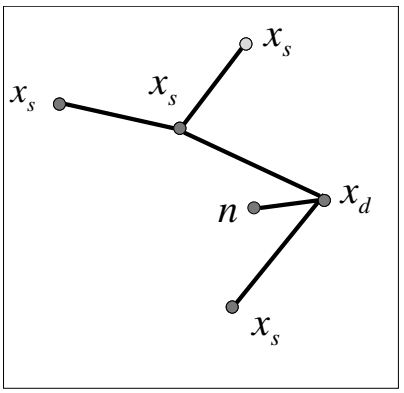

図 2: 辺追加後

くなってしまう．これらの問題を解決するために，辺の削除は存在する．

以下それぞれのステップについて述べる．

3.1.1 距離計算

距離計算のステップについて説明する．入力 n が与えられた時，n と既存頂点群との距離計算 (ユークリッド距離) を行なう．計算した距離は全て距離テーブルに格納しておく．以下に n と任意の頂点 x 間の距離計算式を記述する．式中の $dis(n,x)$ は n と任意の頂点 x との距離を表わす．

n と x の距離計算式. d 次元の場合

$n = (n_0, n_1, n_2, ..., n_{d-1})$

$x = (x_0, x_1, x_2, ..., x_{d-1})$ とすると

$$dis(n,x) = \sqrt{\sum_{i=0}^{i=d}(n_i - x_i)^2} \qquad (1)$$

計算コストはパターン空間が d 次元のユークリッド空間のとき，既存頂点数が N 個の頂点に対し，$O(dN)$ となる．

3.1.2 辺追加

辺追加のステップについて説明する．辺を追加しなければ，その入力はスタックノードとなってしまう．例えば頂点入力直後は図 1 のような状態になり，n からはどの頂点にも近づくことが出来ず連結性を満たさない．よって辺追加を行う必要がある．

辺追加は入力 n から既存頂点群へ，逆に既存頂点群から入力 n へ近付けない頂点が無くなるように辺を追加していくアルゴリズムである．図 2 は近づけない頂点の中で最も近い頂点と入力の間に辺を追加した状態である．この状態では n は x_d に近づくことが出来る．このままでは他の頂点には近づくことは出来ないために近づけない頂点を再び計算し，辺を追加するということを続ける．

全距離計算手法辺追加アルゴリズム 1

入力 n，iNFTG の辺集合 E，頂点集合を V とし，頂点 x_i から頂点 x_j に近づく時，x_i がスタックすることを $Stuck(x_i, x_j)$ と表記する．

1. 頂点集合 V の中から $Stuck(n, x_s)$ となる x_s を全て求める．ここで求められた x_s の集合を C とする．
2. 次の式を満たす x_d を求める．
$$x_d = \arg\min_{x \in C} d(n,x)$$
3. $E = E \cup (n, x_d)$
4. 1〜3 を繰り返し $C = \phi$ になるまで繰り返す．

ここまでで n から既存頂点群には近づくことが可能となる．次に既存頂点群から n に近づくことが出来るかどうかを計算する．１〜４のステップだけでは，既存頂点群から n に近づくことが出来るか（スタックノードにならないか）どうかは言えないからである．そこで既存頂点群から入力へ近づけるかどうかを計算し，近づけない頂点群の中で最も近い頂点と入力の間に辺を追加するという作業が必要となる．

全距離計算手法辺追加アルゴリズム 2

1. 頂点集合 V の中から $Stuck(x_s, n)$ となる x_s を全て求める．ここで求められた x_s の集合を C とする．
2. 次の式を満たす x_d を求める．
$$x_d = \arg\min_{x \in C} d(n,x)$$
3. $E = E \cup (n, x_d)$
4. 1〜3 を繰り返し $C = \phi$ になるまで繰り返す．

以上が辺追加のアルゴリズムである．このステップでの計算コストは頂点数 N 個の場合 $O(2N)$ であり，非常に高速であると言える．

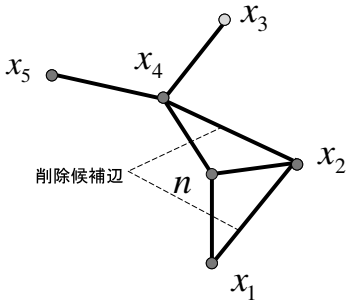

図 3: 削除候補辺

3.1.3 辺削除

辺削除のステップについて説明する．まず削除候補点集合 P を作成する．削除候補点集合とは辺追加ステップで求めている x_d の集合のことである．(図 3 参照) この集合内の点の組み合わせで辺が存在する場合, その辺を削除候補辺と呼び，削除候補辺の集合を削除候補辺集合 U と呼ぶことにする．辺の削除で考えるのは削除候補辺の必要性のみをチェックすれば良い．

全距離計算手法辺削除アルゴリズム

削除候補点集合を P とする．

1. 初期状態として $P = \phi$ とする．
2. 全距離計算手法辺追加アルゴリズム 1，2 で求めた x_d の頂点集合を P とする．
3. 頂点集合 P の中から 2 頂点 x_i，x_j を選択し，全ての組み合わせで次の式を計算する．
   $if(E \in (x_i, x_j))$ then$[U = U \cup (x_i, x_j)]$
4. 削除候補辺集合 U の中から辺を一つ選択する．これを仮に (x_i, x_j) とし，$E = E - (x_i, x_j)$ をする．
5. 頂点集合 V の中に存在する全ての頂点 x_s について $Stuck(x_i, x_s), Stuck(x_j, x_s)$ が 1 つでも成り立つならば $E = E \cup (x_i, x_j)$ をする．
6. U の集合の全ての辺について 4〜6 の手順を繰り返す．

以上が辺削除のアルゴリズムである．このステップでの計算コストは頂点数 N 個，削除候補辺数が M 本の場合 $O(2NM)$ である．削除候補辺数は高次元空間でも多数存在しないため，コストは殆ど辺追加と変わらない．

図 4 に iNFTG の構成の様子を示す．頂点数 6，次元数 2 で構成した iNFTG が (a) の図である．この iNFTG を初期状態として，入力が与えられた図が (b) である．この (b) の状態で，入力頂点と既存の頂点との距離を計算している．(c) は辺追加をした後の図である，単調性，連結性は満たされているが，極小性は満たされていない．そこで (d) は辺削除を行なった後の図であり，辺削除することにより，極小性が完全に満たされているわけではないが，辺を減らすことが出来ている．

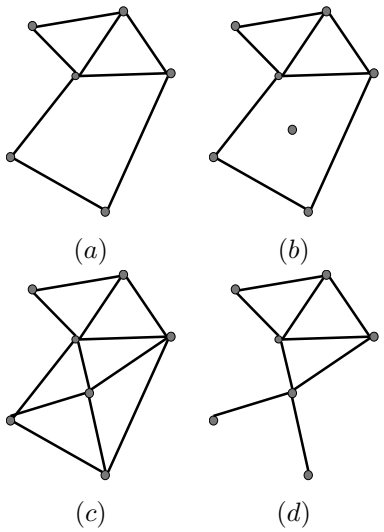

図 4: iNFTG 構成の流れ

3.2 擬似ソート法

擬似ソート法とは，ランダムに追加されていく頂点を擬似的にソートし，辺削除をより正確にするための手法である．図 5 は 3.1.1〜3.1.3 小節で述べたアルゴリズムで構成した iNFTG であるが，辺が密集しているところや，辺の長さが不自然に長いものが存在しており明らかに極小性を満たしておらず，辺削除の条件が完璧ではないことがわかる．

極小性が満たされていないということは辺数が多くなり，頂点の平均分岐数が増えてしまい，NstFT を行なう場合，コストが大きくなるという問題が生じる．逆に辺数が少なければ少ない程，実際に NstFT が高速になり，また iNFTG 全体の構成時間も抑えることが出来るというメリットも存在する．

ここで述べるソートとは，ランダムに追加されていく頂点を，パターン空間中の原点との距離でソートし，順番に入力していく方法である．図 6 は図 5 の頂点群をパターン空間中の原点との距離でソートして頂点追加して構成した場合の iNFTG である．図 5 に比べて図 6 の辺の数が少なくなっている．このことからソートは有効だと言える．

3.3 擬似ソート法の適用

実際の頂点追加はソートされているわけではなく，ランダムに追加されるので擬似的にソートし

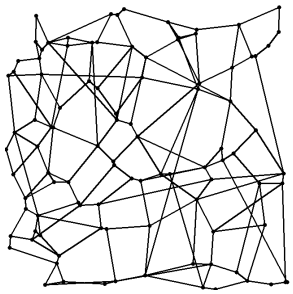

図 5: 擬似ソート無しの iNFTG 概形

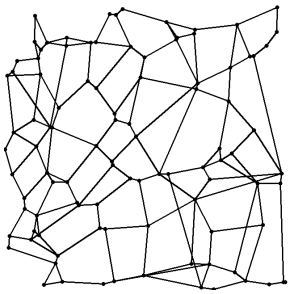

図 6: 擬似ソート有りの iNFTG 概形

て頂点を追加しなければならない．辺追加の手順で，辺の追加先の頂点はソートした時に先に入力されていた頂点か後に入力されるべき頂点なのかという二種類に分けることが出来る．後者の場合，追加された辺というのはソートすると実際には追加されることの無い辺であり，これは別の頂点が点が入力されていく上で必要でない可能性が高い．そこでこの点を擬似削除候補点とし，追加した辺を擬似削除候補辺として記憶し，頂点入力毎にこの辺集合の必要性をチェックすれば良い．

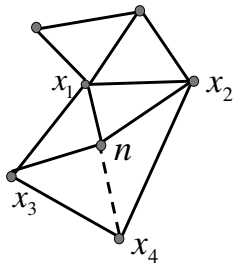

図 7: 擬似削除候補辺

図 7 は図 4 の (c) で追加した辺の中で，擬似削除候補辺 (点線で示す) にあたる辺を示したものである．ソートした順番で頂点を入力していった場合，x_4 は入力 n の後に追加されるべき頂点である．となると，点線で示した辺というのは実際には存在しない辺となる．この辺を擬似削除候補辺とし，各頂点入力毎に必要性をチェックすれば良い．以下この手法を適用したアルゴリズムを示す．

擬似ソート法アルゴリズム

このアルゴリズム内では，パターン空間内原点と任意の頂点 x までの距離を $|x|$ で表わすことにし，擬似削除候補点集合を S とする．

1. 全距離計算手法辺追加アルゴリズム 1，2 で求めた x_d について，$|x_d| > |n|$ を満たす x_d の集合を S とする．
2. 頂点集合 S の中から 2 頂点 x_i，x_j を選択する．
3. E の中に (x_i, x_j) が存在するならば，$U = U \cup (x_i, x_j)$ とする．これを E 内の頂点全ての組み合わせについて行なう．
4. U の中から一辺 (x_i, x_j) を取りだし，$E = E - (x_i, xj)$ をする．
5. 頂点集合 V の中に存在する全ての頂点 x_s について $Stuck(x_i, x_s), Stuck(x_j, x_s)$ が 1 つでも成り立つならば $E = E \cup (x_i, x_j)$ をする．
6. U の集合の全ての頂点の組み合わせについて 4〜5 の手順を繰り返す．

3.4 iNFTG 構成アルゴリズム 2 - 辺分割手法

本節では，iNFTG の二つめの構成アルゴリズムとして辺分割手法について述べ，その将来性について考察する．

3.4.1 辺の垂直二等分線によるグラフ分割

辺分割手法とは 3.1 節で計算した距離計算を可能な限り減らすために考え出されたアルゴリズムである．全距離計算手法の問題点はこの距離計算コストにある．パターン空間が d 次元のユークリッド空間のとき，既存頂点数が N 個の頂点に対し，$O(dN)$ となるが，それでも高次元，では頂点数が増えれば増える程時間がかかってしまう．

そこで本節では頂点間の距離を直接計算して，距離の大小で近づけるかどうかを判定するのではなく，辺に着目し，辺の向きから近づけるかどうかという判定を行う手法を提案する．

3.4.2 辺分割アルゴリズム

ある任意の辺 (両端の頂点を仮に p，q とする) の垂直二等分線を考え，空間を二分割することを考える．頂点 p が存在する空間を P，頂点 q が存在する空間を Q とした場合，頂点 p は空間 Q の中にある頂点ならば，頂点 q を辿って近づけるということが言える．

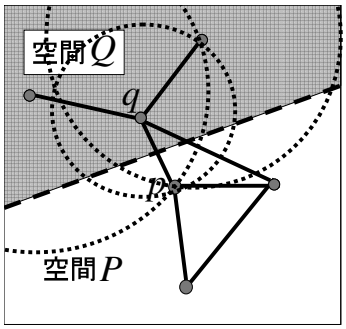

図 8: 分割図

これは空間 Q 内の任意の頂点を中心とし，p までの距離を半径とする円を描いた時，円内に必ず頂点 q が入っていることから判る (図 8)．これらの性質を用いて，全ての辺それぞれがグラフを分割し，辺そのものに各頂点がどちらの領域に入っているか (空間 P,Q どちらに属するか) という情報を記憶させれば，iNFTG の辺追加，辺削除時に距離を使用せずに構築することが出来る．

記憶すべき辺分割情報は着目中の辺の垂直二等分線を考え，それぞれの頂点がどちら側の領域にあるかという情報である．

しかし，現在では，効率的な分割情報作成アルゴリズムは考案されておらず，距離を計算して作成している．そのために，構成時間は全距離計算手法よりも遅くなっている．

辺分割アルゴリズムでも，入力が来ると辺の追加を行ない，次に辺の削除を行なうという流れは変らない．

以上の考え方を導入したアルゴリズムを以下に記す．

辺分割手法アルゴリズム (辺追加部分)

n がスタックノードとなる頂点群を V' とする．

1. 初期状態として $V' = V$ とする．
2. n をターゲットとして任意の頂点から NstFT を行ない，n に近い頂点 x_d を求める．ただし，NstFT 中に辿る頂点は V' の中に入っているものとする．
3. $E = E \cup (n, x_d)$
4. 辺 (n, x_d) の分割情報を作成し，頂点 n 側の領域に属する頂点群を W とする．
5. $V' \cap W$ を満たす頂点群を新たな V' として，もし V' が ϕ なら終了．そうでないなら 2 へ戻る．

2 のステップでは n から近づけない頂点群 V' の中でも最も近い点を出力し，3でその頂点と入力の間に辺を追加している．そして 4 で追加した辺の分割情報を作成し，最後の 5 で n がスタックノードとなる頂点群 V' を計算し繰り返している．

図 9 は，辺追加の一例である（初期状態は図 1）．近づけない頂点 x_2 と結び，その垂直二等分線を考える．近づけない頂点群を V' を計算し，その中で近い頂点を NstFT で出力して辺を追加する，というのを繰り返す．

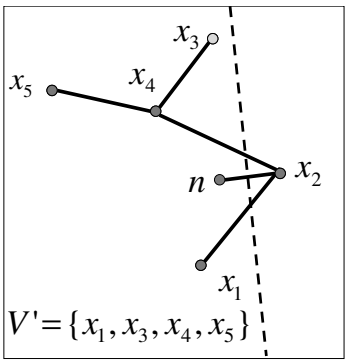

図 9: 辺分割手法による辺追加例

辺の追加終了後，続いて辺の削除を行なう．全距離計算手法の場合は着目中の削除候補辺の端点を x_i, x_j とすると x_i, x_j それぞれが全点に近づけるかどうかを計算したが，辺分割手法の場合は着目中の頂点を端点とする辺集合を求め．その辺集合の辺だけで全ての頂点に近づけるかどうかをチェックする．以下にこの手法を用いたアルゴリズムを記す．

辺分割手法アルゴリズム (辺削除部分)

削除候補点集合を P とする．

1. 初期状態として $P = \phi$ とする．
2. 辺分割手法で求めた x_d の頂点集合を P とする．
3. 頂点集合 P の中から 2 頂点 x_i，x_j を選択する．ただし一度選択した辺は除く．
4. もし E の中に (x_i, x_j) が存在するならば，以下の手順を行なう．そうでないなら 3 に戻る．
5. $E = E - (x_i, x_j)$
6. x_i を端点とする辺集合 S，x_j を端点とする辺集合 T を求める．
7. S 内にある辺のみで x_i は全点に近づけるかどうか判定．近づけるなら以下の手順を行ない，そうでないなら，$E = E \cup (x_i, x_j)$ をした後に 3 に戻る．
8. T 内にある辺のみで x_j は全点に近づけるかどうか判定．近づけるなら以下の手順を行ない，そうでないなら，$E = E \cup (x_i, x_j)$ をした後に 3 に戻る．

9. P の集合の全ての頂点の組み合わせについて 4〜6 の手順を繰り返す.

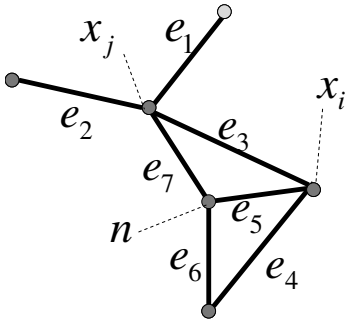

図 10: 辺分割手法による辺削除例

図 10 は辺分割手法による辺削除例である．削除候補辺の選択基準は全距離計算手法と同じであるので，ここでは辺 e_3, e_4 が削除候補辺にあたる．仮に e_3 の両端を x_i, x_j とすると x_i は e_4, e_5 で全ての頂点に近づけるかどうか見る．同様に x_j は e_1, e_2, e_7 で全ての頂点に近づけるかどうかを見て，両方真なら辺 e_3 は削除出来るといえる．

辺分割手法の計算コストは任意の頂点が結ばれている辺数を $B(G)$ 本とした場合, $O(B(G))$ であり，頂点数ではなく，辺の本数に依存している．しかしながら，現時点では辺情報作成は効率のよいアルゴリズムが考えられてないために，現時点での構成コストは頂点数 N とした場合，$O(2B(G)dN)$ となっている．

4 比較実験と考察

我々が提案している iNFTG 構成アルゴリズムの性能評価をするために，評価実験を行なった．

実験には一様分布の乱数によって生成した人工データを用いた．データ数は 1500，次元数を 1000 として，NFTG と iNFTG の比較，また，iNFTG でも全距離計算手法によって生成した iNFTG と辺分割手法によって生成した iNFTG の性能評価も行なった．

4.1 NFTG と iNFTG の検証

本節では iNFTG と NFTG の比較実験，検証を行なう．始めに構成したグラフが構成可能かどうか検証し，構成時間比較，擬似ソート法の有用性について考察する．

4.1.1 グラフ比較

始めにこれまで述べて来た iNFTG が実際に高次元空間，多数のデータ数でも構築可能かどうか検証する．

図 11 は今まで述べて来た手法で NFTG を構成した場合の辺数を示している．sort on,sort off は全距離計算手法で辺を追加した後，擬似ソート法で辺を削除したかどうかを表わしていて，divide edge は辺分割手法で構成した iNFTG，そして NFTG は既存の NFTG を表わしている．この図から，全ての iNFTG は NFTG と殆ど辺数が変らないことが判る．NFTG の辺数とほぼ同じと言うことは，実際にグラフサーチに使用可能であると言える．

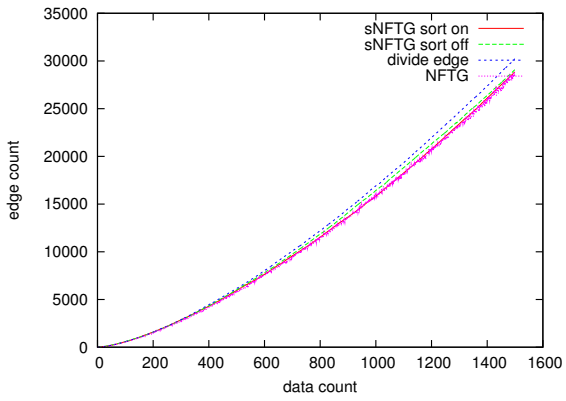

図 11: 構成方法の違いによる辺数比較

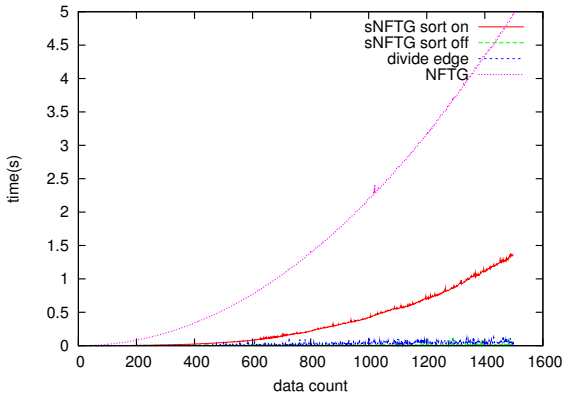

図 12: 構成方法の違いによる構成時間比較

4.1.2 グラフ構成時間比較

図 12 は実際に一点ずつ頂点を追加していった場合の iNFTG 構成時間をプロットした図である．最も時間がかかっている曲線は既存の NFTG 構成方法による構成時間である．それらに比べて提案手法は非常に高速に構成出来ていることが判る．

最も速いのは全距離計算手法 (擬似ソートを用いない) で，次が辺分割手法．そして 1500 点目で 1.2 秒程度掛かっているのが全距離計算手法 (擬似ソートを用いる) である．擬似ソートを用いない全距離計算手法と辺分割手法二つのみの構成時間をプロットしたものが，図 13 である．振動が見られるが，ほぼ線形に時間が増えていることが観測される．若干辺分割手法を用いた方が遅くなっているが，これ

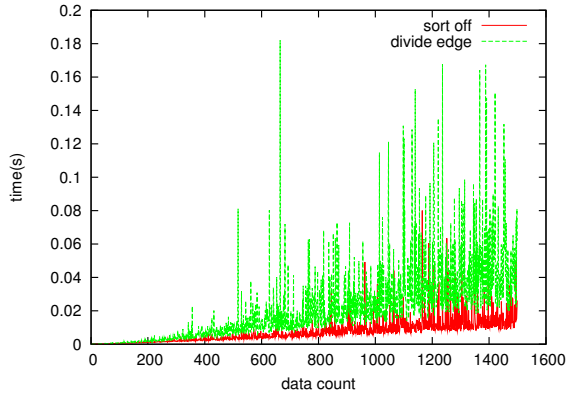

図 13: 全距離計算手法 (擬似ソート無し) と辺分割手法の構成時間比較

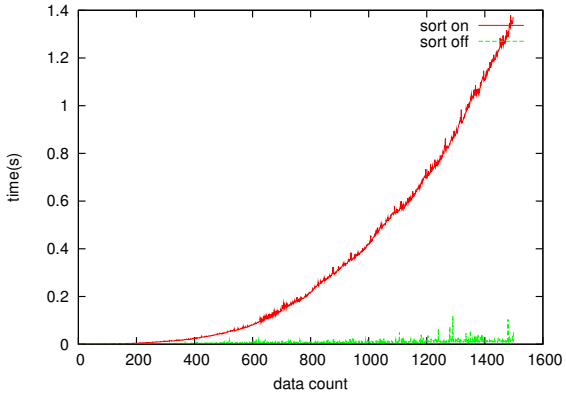

図 14: 擬似ソート有無による構成時間比較

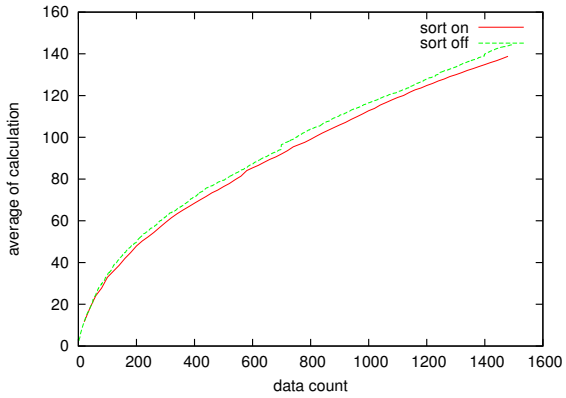

図 15: 擬似ソート有無による平均距離計算回数

は辺情報作成に全ての頂点間の距離計算を行うという非常に効率の悪い作成方法であるために，余分に時間がかかっている．

効率の良い辺情報作成アルゴリズムが開発されればこの差は更に縮まり，全距離計算手法を用いなくても iNFTG を構成することが可能である．

4.1.3 擬似ソート法の有用性

本節では擬似ソート法の有用性について検証をする．

図 14 は全距離計算手法で，擬似ソートをした場合としなかった場合で，データ数を増やしていった場合の構成時間比較である．擬似ソートの有無ではかなりの構成時間差がある．続いて図 15 は実際に NstFT を行なった時の平均距離計算回数である．この図を見ると数回の差は見られるが，しかし致命的に回数が増えているということは言えない．つまり，実際にグラフを使う場面において擬似ソートの有無による影響は少ないと考えられる．擬似ソート法を適用すると，グラフ辺数が少なくなり，より極小性を満たすグラフが作成することが出来るが，図 14 のように構成時間が膨大になるために，擬似ソート法はあまり良い方法だとは言えない．

5 結論

本論文では，高次元空間でも構成可能な NFTG を逐次的に追加されていく入力に対して高速に構成することが可能な iNFTG の構成アルゴリズムを提案した．

iNFTG の構成アルゴリズムの 1 つである全距離計算手法は全ての入力間の距離を計算し，必要最低限のグラフ構築（辺の追加，辺の削除と 2 つのステップ）により構成するものである．更に辺の削除をより正確にするための擬似ソート法を提案した．

2 つめの構成アルゴリズムの辺分割手法では，頂点間の距離を計算して構成していくのではなく，辺そのものの情報から空間を分割し，既存の入力頂点がどちらの領域に入っているかという情報のみで構成することを提案し，距離計算をしなくても構成出来るという点から，より良いものであると推測出来ることを論じた．

それぞれの構成アルゴリズムを用いた比較実験を行ない，現在最も良い方法は全距離計算手法で，尚且つ擬似ソート法による辺削除を行なわない方法が最も高速で，良いであろうと考えられる．

今後は辺分割手法の最も問題となっている辺情報作成アルゴリズムについてより良い計算アルゴリズムを検討していく予定である．

参考文献

[1] 坂垣内，和田，加藤：“ 近さ優先探索グラフを用いた実時間対象追跡・認識 ”，画像の認識・理解シンポジウム (MIRU)，pp.16-23(2005).