対話的なネットワーク可視化分析のシェーダー実装による高速化

高野 陸　脇田 建

社会ネットワークの可視化はデータの大規模化に伴い，視覚的に煩雑になる点とシステム描画が遅くなる点の二つの問題を抱える．対話的可視化システムである Social Viewpoint Finder (SVF) は過去の実装において視覚的煩雑さをフィルタリングで解決した一方，システム描画の遅さは依然とした問題であった．システム描画の遅さの原因として，描画のための CPU-GPU 間通信量が多い，CPU で負荷の高い計算を行なっているなどが挙げられる．そこで本研究では CPU-GPU 間通信を軽量化するデータ配分や，負荷の高い計算を計算シェーダーで並列に実行するなどの工夫をすることでシステムの高速化を図った．過去の実装に比べ，本研究での実装はシステム全体を通して 20 倍程度の処理速度の向上を達成した．

1　はじめに

情報可視化の手法は，様々なネットワークを分析する一つの手段として重要である [5]．しかし，世の中に存在するネットワーク，特に社会ネットワークの多くは大規模であることが多く，可視化システムにとって大規模なネットワークを分析できるスケールを持つことは難しい．またスケール可能な可視化システムであっても高度な視覚的分析のタスクに対応出来るものは少ない．

ネットワーク可視化には，ネットワークを一枚の静止画で表す静的可視化と [3] [2] [9]，可視化結果を対話的な操作によって観察できる対話的可視化がある [10] [8] 静的可視化システムは大規模なネットワークに対応できるが複雑なタスクは難しく，これまでの対話的可視化システムは複雑なタスクが可能であるが大規模なネットワークに対応しきれないと特徴付けられる．

静的可視化システムではネットワークを一枚の静止画で表現するが，大規模化につれて問題が生じる．ネットワーク全体を描画すると，頂点と辺の重なりなどによる視覚的な煩雑さがネットワークの構造把握を妨げ，視覚的な分析が難しくなる．

対話的可視化システムでは，対話的に可視化結果を操作できるためネットワークの構造をより深く理解できる．高度な対話性ならば高度なタスクを実行可能であるが，相応の計算コストが掛かるため大規模なネットワークを可視化する性能を達成することが難しく，逆に簡素な対話性では大規模化が可能でも，高度なタスクに対応できないという問題がある．

本研究では，高度な対話性を提供しながら大規模なネットワークを可視化できるシステムを実装した．使用したシステムは高見らが提案した *Social Viewpoint Finder* (*SVF*) [12] という対話的可視化システムである．高見らの実装ではクラスター構造の発見に役立つことを示していたが，大規模なネットワークの可視化には難を残していた．本研究の実装で SVF の高度な対話性を損ねることなく，描画と計算の工夫によってシステムの性能を向上させ大規模なネットワークの可視化を実現する．

2　関連研究

ネットワーク可視化システムで静的なものと対話的なものを例示する．また，それらの利点や対応できるネットワークの規模，システムの問題点にも触れる．

Riku Takano, Ken Wakita, 東京工業大学 情報理工学院.

2.1 静的可視化

ネットワーク可視化の手法の一つとしてノードリンク図が存在し，これはネットワークにおける人や事物を円，そのリンク関係を線で描く手法である．可視化アルゴリズムの研究はその円と線の適切な配置を求める分野であり，可視化された配置を**埋め込み**と呼ぶ．

ネットワーク可視化のアルゴリズムの計算量は頂点数や辺数について線形ではないため，大規模化と共に計算コストが急激に増大する．HDE [3] という可視化アルゴリズムでは，百万頂点のネットワークの埋め込みの作成を約 2 分で行うことができる．他にも，Gansner らの Maxent-Stress Model [2] や，Noack の Lin-log [9] などが大規模なネットワークの可視化を達成している [7]．

静的可視化は大規模なネットワークに対応できても，視覚的な分析を十分に行うことが困難であるという問題点がある．これは複雑なネットワークの構造が一目でわかる静的な配置を求める難しさと，大規模なネットワークで起こる頂点や辺の重なりなどによる可視化結果の視覚的な煩雑さが原因である．

2.2 対話的可視化

対話的可視化とはユーザーとの相互作用的に可視化を行う手法である．頂点のラベルを表示する操作やアフィン変換，フィルタリングといった簡素な処理のほか，エッジバンドリングなどの複雑な処理を対話的に行う機能が例として挙げられる．

静的な可視化では最終的に出来上がった可視化結果で分析を行うため，過程となる描画の遅速は問題とならなかった．しかし対話的可視化の場合は対話によってネットワークの分析を行うため描画速度は重要である．対話操作の応答は操作の種類によるが遅いもので１秒，速いもので３０ミリ秒以内の完了が求められ，対話操作に要する計算時間がこれを超えるとユーザーの対話体験を損なう．そのため対話的可視化はシステム描画がある程度高速であることが求められる．

Panagiotidis らによる可視化 [10] では拡大や縮小などのアフィン変換や，頂点と辺にマウス操作で手を加えるといった対話的機能を提供している．この可視化では 40 万頂点と 339 万辺のネットワークについて，対話操作中も 5.1FPS[†1] を保ち頂点や辺の選択を 1.83ms で実行することができた．Muelder らの SFC [8] はズーム機能を提供しており，他にも大規模ネットワークの対話的な可視化が研究されている．

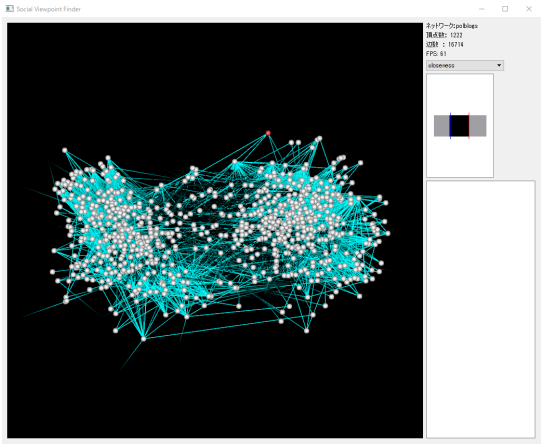

図 1　SVF による米国の政治ブログの可視化

これらのような対話的可視化はアプリの応答から導かれる簡素な対話機能を備えている．しかしクラスター構造の発見といった高度なタスクに対応することは難しく，アプリの応答の他にネットワークの構造を利用する等の高度な対話性が要求される．

3 Social Viewpoint Finder

高見らの提案した SVF は社会ネットワークを対話的に可視化し，視覚的な分析を可能にするシステムである．頂点をドラッグすることで高次元回転による視座の変更を対話的に行うことができ，またフィルターとの併用によりネットワークのクラスター構造を探索することができる．

3.1 システムの概要

Torgerson-Kluscal-Seeri **法** (*TKS* **法**) は**古典的多次元尺度法** (*CMDS*) [14] [11] [13] の結果から主固有ベクトルからなる基底空間を平面に射影する．これに対し，細部が提案した *Active Graph Interface*(*AGI*)

†1　FPS は画面描画性能の単位．5.1FPS は秒間に 5.1 画面を描画できる性能を意味する．逆に画面更新に要する時間は約 200ms となる．

は正のすべての固有値を合成して得られる基底空間を平面に射影する一般化を前提とした対話機能 [4] [12] である．AGI は射影された平面上の対話操作によって高次元回転を行い視座を対話的に変更する機能を提供する．

SVF では対話操作に AGI を用いており，頂点数 n のネットワークを可視化するとして手法の説明を行う．先ずはその描画の初期配置を求める計算の説明をする．この計算をまとめると次の 3 つのステップからなる．

1. CMDS による高次元配置 L_{hd} の導出
2. 細部の方式に従って初期射影行列 P を構成
3. 射影された平面上の頂点の配置 L_{ld} を計算

最初のステップは，ネットワークの隣接関係からの全対グラフ論的距離行列とそこから導出された行列の固有値分解が主要な計算である．高次元配置は $(n \times d)$ 次元の行列であり，d は高次元配置上の頂点の次元数である．大規模なネットワークを扱う際の効率化のために，描画結果への貢献度が低い頂点の次元を省略する．具体的には，頂点の次元 d を最大 500 までとして高次元配置 L_{hd} を求める．

次に細部の方式に従って初期射影行列 P を構成する．行列 P の次元は $(d \times 2)$ である．

最後に射影された平面上の頂点の配置 L_{ld} を，$L_{hd} \cdot P$ の行列積の計算によって求め，描画の初期配置とする．行列 L_{ld} の次元は $(n \times 2)$ である．

AGI の対話機能はドラッグ操作によって開始される．頂点をドラッグして移動させた座標について，移動後の位置に射影するような新しい射影行列 P' を求める．射影行列は適切な更新のための制約式を解消することによって求められる．この射影行列の更新は埋め込みの高次元的な回転を表しており，回転によってドラッグ後の頂点の配置 L'_{ld} が計算される．この対話操作の計算をまとめると次の 3 つのステップからなる．

1. 選択された頂点のドラッグ後の座標を取得
2. 制約式を解消し新たな射影行列 P' を計算
3. 新しく射影される頂点の配置 L'_{ld} を計算

対話操作の中でドラッグにより頂点を動かす前に，そもそも頂点の選択を行う必要がある．つまりユーザーがクリックした座標直下に描画された頂点の識別番号を同定する処理が対話操作の初めに必要となる．その後に選択された頂点をドラッグし，ドラッグ後の座標によって以下の計算に続く．

制約式は 8 元非線形連立方程式であり，高見らの実装と本研究の実装のどちらも L-BFGS [6] によって近似解を求めている．最適化によって $(d \times 2)$ 次元の新しい射影行列 P' を得たら，新しく射影された平面上の頂点の配置 L'_{ld} を，$L_{hd} \cdot P'$ によって求める．これらの計算はドラッグ操作のフレーム毎に発生する．

SVF には他にも対話機能が備わっており，その中の一つとして中心性によるフィルタリング機能が備わっている．ネットワークの頂点と辺について**中心性**という種々の重みが存在し，この中心性の種類と重みの閾値を決定することで頂点と辺の描画を制限することができる．これを対話的に行うことで，種々の指標に基づいたネットワークの要約を探索することができ，構造理解の一助となる．また描画する頂点と辺を削減することで描画処理を軽くするという意味もあり，高見らの実装ではフィルタリングなしには大規模なネットワークの可視化は不可能であった．

SVF は社会ネットワークの視覚的な分析を効果的に実現できたが，対話的な可視化システムとして運用可能な規模の限界は，約 4,000 頂点と 45,000 辺程度のネットワークであった．高見らの実装において，これ以上の規模では実用性を保つことは難しかった．

3.2 高見らの実装の性能における問題点

高見らの実装において言語は C++を使用し，GLUI によって古典的 OpenGL を使用していた．SVF が大規模なネットワークを可視化する際に，システム描画が遅くなる原因となる実装を描画と計算の面で考える．

描画については，描画フレーム毎に頂点と辺のデータが GPU へ送られる．描画には座標と色を送るのでデータ量は (28 × 頂点数 + 40 × 辺数) B に及ぶ．大規模なネットワークにおいてこの通信の負担は大きくなる．特に頂点と比べて辺は数が多く，その数は数倍にも登るために描画の遅延の一因となっていた．描画トランザクションが細分化されていた点も原因の

図 2　高見らの実装において Enron データを可視化した際の SVF のシステム概観

一つである．

計算については次の三つの点に注目した．第一に射影計算における行列積の計算である．これはドラッグ操作によりリアルタイムに実行されるので，この計算の遅延はシステム描画の遅延につながる．頂点の配置 L'_{ld} を求める $L_{hd} \cdot P'$ の行列積での計算量は大きく，L_{hd} の次元が $(n \times d)$，P' の次元が $(d \times 2)$ として，計算量は $O(d \times n)$ になる．CPU 上での LAPACK[†2] の利用によってある程度高速な並列計算がされていたものの，可視化するネットワークが大規模になるにつれて遅延が目立つようになっていた．

第二にフィルタリングである．これは CPU で全ての頂点について選ばれたフィルターの閾値に照らし合わせることで描画するかを決めており，描画フレーム毎にフィルタリングの処理を逐次行っていたため頂点や辺の数に比例して計算コストがかさんでいった．

第三に頂点選択の処理である．クリック時の座標にマッチする頂点を CPU 上で全探索しているため，大規模なネットワークにおいて頂点選択の処理時間が大きくなり，クリックに対する反応が遅れてユーザーの快適さを損う問題があった．

図 2 は 33,696 頂点と 180,811 辺をもつネットワークである Enron データ [†3] を可視化した際のメモリ使用量や通信量を大まかに勘定したものである．この実装では巨大なデータをほとんど CPU で処理し，また描画フレームごとに少なくとも 8.1MB のデータを通信する必要がある．PCI バス [†4] の転送速度が 1 レーンあたり 1GB/sec として，描画フレームごとのデータ転送時間は最低でも 8.1ms と計算できる．[†5]．当然ながら描画フレームで転送するデータは他にも存在し，描画トランザクションの細分化も合わさって大きな遅延を生むと推測できる．

高見らの実装で大きな問題点であった描画性能の不足に焦点を当てて，本研究ではそれを踏まえた実装の改善を行う．

4　高速化を施した SVF の実装

本節では，3 節で触れた実装の問題点を改善し，描画性能と計算性能を高めるために次の 5 つの工夫を行なった．本研究で実装に使用した言語は Python であり，PyOpenGL によって現代的 OpenGL を使用した．

また，本研究の実装によって 3.2 での性能がどの

†2　http://www.netlib.org/lapack/
†3　https://www.cs.cmu.edu/ ./enron/
†4　パソコン内部の各パーツ間を結ぶデータ伝達路．高見らの実装で使用していた PCI バスは不明のため，本研究で用いた PCI バスの性能で計算する．
†5　データ量は 8.1[MB/frame]．PCI バスの転送速度が 1000[MB/s] より，データ転送にかかる時間は「データ量 [MB/frame] / (転送速度 [MB / s] * 1000[s/ms])」より 8.1[ms/frame] と計算できる．

程度改善されるのかの見通しを述べる．

4.1 本研究で実装した５つの工夫

1．役割に応じたデータの配置

高見らの実装ではすべてのデータを CPU 上に配置し，必要なデータを都度 GPU に送信していた．この通信は大規模なネットワークを可視化する際に無視できない大きさになるため，本研究では最小限の CPU-GPU 間通信に抑えるための効率的なデータの配置を行なった．例えば，頂点を描画フレーム毎に送るのは手間であるので初めから GPU に頂点座標を配置する，並列計算に用いるデータを初めから GPU 上に出来るだけ配置するなどである．これにより CPU 上のメモリを節約し，また CPU-GPU 間の通信量を減らすことが目的である．

2．描画手法の改善

高見らの実装では辺の描画のために端点の座標と色を与えており，一つの座標が 12B，色が 16B であるために一つの辺のデータ量は 40B であった．頂点の描画のために頂点座標は既に送られているため，辺のために再び同じ座標を送るのは二度手間である．よって本研究では辺の端点の座標に頂点のインデックスを与えることでデータの軽量化を行なった．一つの頂点インデックスは 2B であるために一つの辺は 4B で表せるようになり，辺のデータサイズは 1/10 に削減できる．また，高見らの実装では頂点と辺の色も CPU から送っていたが，頂点 ID から頂点シェーダーで色を設定することでこのデータの転送を省いている．そして描画トランザクションも一括して行うことで通信頻度を下げ，更に描画フレームごとに大量に送られる描画命令のデータを減らすことで性能向上を狙った．

3．射影計算における行列積計算の並列化

SVF は射影計算において高次元配置 L_{hd} と射影行列 P' の行列積計算を行う．高見らの実装ではそれを CPU 上で行なっていたが，行列積の計算量は頂点数の増加に応じて激しく増加する．そこで行列積を計算シェーダーで並列に計算することで高速化を図った．本研究の実装では $L_{hd} = (hd_1, ..., hd_n)^T$ として，$L_{hd} \cdot P' = (hd_1^T \cdot P', ..., hd_n^T \cdot P')$ について，一つのスレッドごとに $hd_i^T \cdot P'$ を計算させた．

4．フィルタリング処理の並列化

全頂点について，ユーザーが選択している中心性のスコアと選択されている閾値を GPU 上で保持する．これにより頂点シェーダーが並列描画を実行する際，同時に頂点の中心性のスコアとフィルターの閾値を比較し，フィルタ対象となる頂点の描画を省いている．

5．頂点選択処理の並列化

CPU で取得したクリック座標を GPU に送信し，それを参照して画素シェーダー上で全頂点を並列に探索する．クリックされたと判定された頂点の ID を CPU に送り，どの頂点が選択されたかを知ることができる．

4.2 本研究の実装における性能

図 3 は本研究の実装において Enron データを可視化した際のメモリ消費量や通信量を大まかに勘定したものである．

描画フレームごとに送られるデータは軽量化された辺だけであり，また一括した描画トランザクションのために描画による遅さは改善されたと考えられる．描画フレームごとに送られるデータ量は 0.7MB として，PCI バス †6 の性能は 1 レーンあたり 1GB/sec として，描画フレームごとのデータ転送時間は 0.7ms と計算できる．単純に高見らの実装に比べてこの転送だけでも 1 1 倍速くなった．他にも様々な計算を行うために実際の時間は遅くなるが，計算自体にも工夫を加えて負荷を軽減しているため全体を通してシステム描画の高速化が見込める．特に，高見らの実装で問題であった描画トランザクションの細分化を解消しているため，その点でもデータ転送量が大きく抑えられていると考えられる．また描画フレーム以外での通信は頻度が少なく，データ量も多くないために現状では

†6 PCI Express x16 Gen3 を使用．1 レーンあたりの実効データ転送速度は片方向 1.0GB/sec.

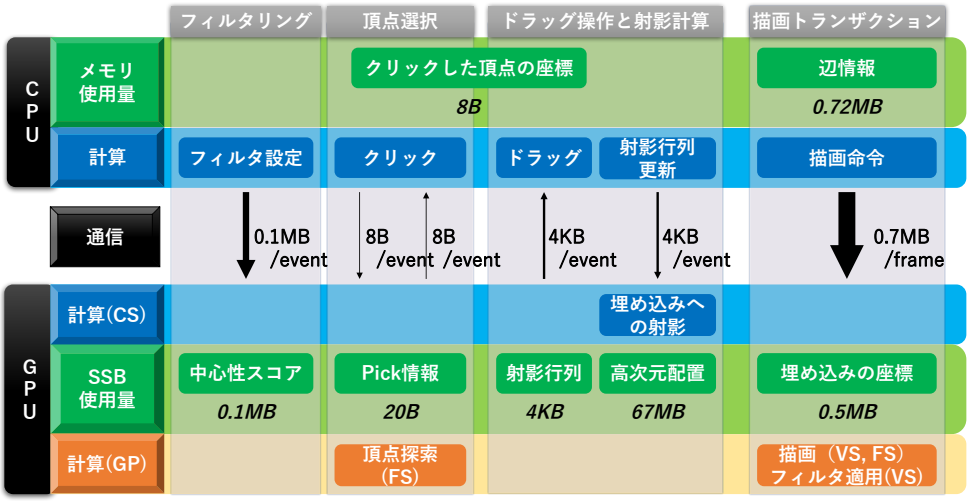

図 3　本研究の実装において Enron データを可視化した際の SVF のシステム概観

問題視していない．

計算については，高見らの実装での計算の一部を並列計算に置き換えた．計算の多くは小さいネットワークの可視化においては高速に行われるため違いがわからないが，大規模なネットワークでは大きな違いが生まれると考えられる．高見らの実装のように CPU で計算を行うと大規模化に耐えられず計算資源の枯渇によって遅延が生じてしまう．この問題は豊富な資源を持つ GPU を利用することで解決できる．

5　考察

本節では初めに本研究で施した性能向上のための工夫を高見らの実装と比べてまとめる．そして高見らの実装と本研究の実装の SVF で同じネットワークデータを用いて性能比較を行う．

5.1　本研究における改善点のまとめ

4 節で述べた本研究の実装における工夫について，3.2 で述べた高見らの実装との違いをまとめると表 1 のようになる．

上記の性能評価において，本研究の実装が高見らの実装を上回った大きな理由として次の二つが挙げられる．

まず描画の改善が一つの理由である．描画のために大量のデータを毎フレームごと GPU に送信していた問題はデータの効率的な配置によって改善され，辺のデータの重さは同等の意味をもつ他の表現を用いることで軽量化し，描画トランザクションをひとまとめにした．これらの貢献によって描画のコストを大きく軽減したことが性能向上の一因と考えられる．

もう一つは，射影計算における行列積の並列化である．射影計算は SVF のドラッグ操作に応じてリアルタイムに計算されるため，この計算の遅延はシステム描画の遅延につながる．高見らの実装では計算をアプリケーションで実行をしており，制約式の解消は高速でも行列積の計算は大規模化につれて遅延が目立っていた．本研究では計算性能改善のために行列積の計算シェーダーを用いて並列に計算した．行列積の計算はベクトルの内積の和に分解して並列化を行なったのみであるが，大幅な改善が見られた．この計算を MAGMA[†7] などの高性能な並列計算ライブラリに

†7　http://icl.cs.utk.edu/magma/

表 1　高見らの実装と本研究の実装との比較

比較項目	高見らの実装	本研究の実装
データ配置	全てのデータを CPU 上に配置	役割に応じて CPU/GPU に分割して配置
辺の表現	端点の座標を直接与えて表現	端点の頂点インデックスを与えて表現
行列積	CPU のマルチコア，ベクトル命令を利用して並列計算	GPU の計算シェーダーで実装した並列計算
フィルター	CPU で逐次処理	頂点シェーダーで並列に処理
頂点選択	CPU で全探索	画素シェーダーで並列に頂点を全探索

よって実行すれば更なる性能向上が見込める．

他にもフィルタリングや頂点選択でも改善を行っており，それらの寄与も考えられる．

5.2　性能評価

高見らの実装での SVF の実行環境は CPU が Intel Core i5 (3.1GHz)，メモリ 16GB，グラフィックスは AMD Radeon HD 6970M の iMac (OS X 10.8.5) であり，本研究の実装での SVF の実行環境は CPU が Intel Core i7 (4.0GHz)，メモリ 8GB，PCI Express x16 Gen3，グラフィックスは NVIDIA GeForce GTX 770 の HP ENVY-700-560jp (Windows 10 Home) である．高見らの実装についての性能評価のデータは論文 [12] のデータを一部引用している．

評価のための性能比較に 5 つのネットワークを使用した．ネットワークの名前について，USPol は米国の政治ブログ [1]，4Univ は Twitter における四大学連合のフォロー関係 †8，Math は Wikipedia の数学に関する記事の参照関係 †9，Internet は BGP の構造 †10，Enron は Enron 社のメールデータである †11．

†8　https://twitter.com

†9　https://ja.wikipedia.org/wiki/Category:数学

表 2　高見らの実装と本研究の実装との描画性能の比較

データ	| V |	| E |	FPS (高見らの実装)	FPS (本研究の実装)
USPol	1,222	16,714	57	60
4Univ	1,896	26,183	35	60
Math	3,608	48,315	30	60
Internet	22,463	48,436	15	60
Enron	33,696	180,811	7.5	60

まず高見らの実装と本研究の実装について描画性能の比較を行う．描画速度に関してはすべてのノードとエッジを表示した状態で 1 分間の平均 FPS を計測した．これらの計測結果をまとめたものが表 2 である．

描画の FPS は最大で 60 であるため，本研究の実装では描画に関して良い結果を得られた．高見らの実装では描画だけで精一杯であった Enron データも，本研究の実装では軽々と描画することができた．描画に関してのみ言えば，本研究の実装は更なるスケールアップが見込める結果となった．

次に高見らの実装と本研究の実装について，対話操作を含めたシステムの性能比較を行う．高見らの実装の対話操作の実行速度は計測されていなかったので，次のように希望的な観測からおよその FPS を導出する．対話操作中の FPS を便宜的に FPS' と表記する．

SVF の対話機能ではドラッグ毎に L-BFGS による制約解消と行列積の計算が実行される．対話操作中も描画を行うため，描画にかかる時間も勘定する必要がある．よって対話操作中のおよその FPS は L-BFGS の制約解消にかかる計算時間と行列積の計算時間の和を実行時間 t1[s] とする．この t1 に 1 フレームの描画時間 t2[s] を足したものを対話操作中における 1 フレームの実行時間として考える．対話操作中の他の計算は無視できるものとして考え，1/(t1 + t2) が FPS' であると概算する．

†10　http://www-personal.umich.edu/ mejn/netdata/as-22july06.zip

†11　https://www.cs.cmu.edu/ ./enron/

表 3　高見らの実装と本研究の実装との対話操作の性能比較

	高見らの実装			本研究の実装	
データ	t1[ms]	t2[ms]	FPS'	t1[ms]	FPS'
USPol	1	17	57	1.51	60
4Univ	1	28	35	1.87	60
Math	5	26	33	2.34	60
Internet	193	67	3.8	5.64	58
Enron	379	133	1.9	7.97	50

L-BFGS は高見らの実装では libLBFGS[†12]，本研究の実装では scipy のライブラリを用いて計算している．この計算については本研究で工夫した点はなく，どちらの実装でもおよそ 1ms で計算が終了するため，この実行時間は無視する．よって t1 は行列積の計算として求める．

高見らの実装において t1 は論文 [12] で評価されており，t2 は表 2 から求まる．これにより FPS' を計算し，高見らの実装における対話操作中の性能とする．本研究の実装では，SVF の対話操作を 1 分間行った平均の FPS を計測して FPS' とし，行列積計算の性能比較のために本研究の実装における t1 を個別に計測した．これらをまとめたものが表 3 である．

高見らの実装では 4Univ と Math における行列積の計算時間 t1 が小さいにもかかわらず描画の遅さによって FPS' が低くなっていることがわかる．本研究では描画性能が改善されているので，対話操作中も快適な FPS' を保っていた．

また高見らの実装では Internet と Enron における行列積の計算時間 t1 と描画時間 t2 の両方が負担となり，FPS' を大きく下げていることがわかる．本研究では行列積を並列に計算することで負荷を軽減しており，t1 は高速に実行されている．描画性能の改善と合わさって，対話操作中では高見らの実装と比べ 20 倍程度の FPS' を達成できた．

†12　http://www.chokkan.org/software/liblbfgs/

6　まとめと今後の課題

描画手法と計算手法の工夫により，対話的可視化システムである SVF のスケールアップを達成した．高見らの実装と比べ少なくとも 10 倍の大きさのネットワークについて対話的に可視化を行うことが出来るようになった．

世の中の社会ネットワークのスケールに至ることが今後の課題であるが，現状の SVF は性能の課題とアルゴリズムの課題がある．性能の課題は描画で辺の情報を GPU に配置して描画フレームの通信量をなくす，行列積を高度な計算ライブラリによって計算するなどによってある程度は解消できる見込みである．アルゴリズムの課題は大規模な行列の固有値計算が厳しい課題となっている．数十万規模の距離行列の固有値を求めることは莫大な計算量を要する．SVF の描画アルゴリズムにおける計算を近似的なアプローチに代替するなどが考えられるが，現状で見通しは立っていない．

また，大規模なネットワークを可視化した際に生じる埋め込みの視覚的な煩雑さについても対処が必要である．

謝辞

研究を支えてくださった皆様に深く感謝いたします．

参 考 文 献

[1] Adamic, L. A. and Glance, N.: The political blogosphere and the 2004 US election: divided they blog, *Proceedings of the 3rd international workshop on Link discovery*, ACM, 2005, pp. 36–43.

[2] Gansner, E. R., Hu, Y., and North, S.: A maxent-stress model for graph layout, *Visualization and Computer Graphics, IEEE Transactions on*, Vol. 19, No. 6(2013), pp. 927–940.

[3] Harel, D. and Koren, Y.: Graph drawing by high-dimensional embedding, *Graph Drawing*, Vol. 2528, 2002, pp. 207–219.

[4] Hosobe, H.: A high-dimensional approach to interactive graph visualization, *Proceedings of the 2004 ACM symposium on Applied computing*, ACM, 2004, pp. 1253–1257.

[5] Lee, B., Plaisant, C., Parr, C. S., Fekete, J.-D., and Henry Riche, N.: Task Taxonomy for Graph Visualization, ACM, April 2006, pp. 1–5.

[6] Liu, D. C. and Nocedal, J.: On the limited memory BFGS method for large scale optimization,

Mathematical Programming, Vol. 45, No. 1(1989), pp. 503–528.

[7] Ma, K.-L. and Muelder, C. W.: Large-scale graph visualization and analytics, *Computer*, Vol. 46, No. 7(2013), pp. 39–46.

[8] Muelder, C. and Ma, K.-L.: Rapid graph layout using space filling curves, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 14, No. 6(2008).

[9] Noack, A.: Energy Models for Graph Clustering., *LNCS 2912, Springer*, (2004), pp. 425–436.

[10] Panagiotidis, A., Reina, G., Burch, M., Pfannkuch, T., and Ertl, T.: Consistently GPU-Accelerated Graph Visualization, *Proceedings of the 8th International Symposium on Visual Information Communication and Interaction*, VINCI '15, New York, NY, USA, ACM, 2015, pp. 35–41.

[11] Torgerson, W. S.: Multidimensional scaling of similarity, *Psychometrika*, Vol. 30, No. 4(1965), pp. 379–393.

[12] Wakita, K., Takami, M., and Hosobe, H.: Interactive high-dimensional visualization of social graphs, *2015 IEEE Pacific Visualization Symposium (PacificVis)*, IEEE, 2015, pp. 303–310.

[13] Young, F. W.: Multidimensional scaling, *Encyclopedia of Statisical Sciences*, Vol. 5, Wiley, 1985, pp. 649–658.

[14] Young, G. and Householder, A. S.: Discussion of a set of points in terms of their mutual distances, *Psychometrika*, Vol. 3, No. 1(1938), pp. 19–22.