Split4Blank: ブランクノードを多く含むRDFファイルの分割ツール

Split4Blank: a tool for splitting an RDF file that includes many blank nodes

山口敦子　　山本泰智

Atsuko Yamaguchi and Yasunori Yamamoto

情報・システム研究機構 ライフサイエンス統合データベースセンター
Database Center for Life Science, ROIS

概要: .RDF データセットの大規模化により，トリプルストアに対して，より効率的なロード方法が求められている． 一つの有効な手段として，RDF データのファイルを分割し，並列にロードする方法が挙げられる．しかしながら，ブランクノードはグローバルな ID を持たないため，同じブランクノードが別のファイルに分割されて並列ロードされると，ストア内では一般に同じノードとして扱われない．そこで，同じブランクノードは同じファイルに含むよう，かつ，できるだけ分割後のファイルの最大サイズが小さくなるよう分割するツール Split4Blank を開発した．本稿では，ツール設計の背景となった，問題の定義およびアルゴリズムについて述べる．さらに，実データ及び架空データによるファイル分割の計算機実験を行い，計算時間に関する傾向について議論する．本ツールは https://github.com/acopom/split4blank より利用可能である．

1. はじめに

生命科学分野では多種多様なデータを統合的に扱うために，セマンティックウェブ技術に基づいたデータの記述や公開が推進されてきた．計測機器の大幅な進歩により，これらのデータは日々増大している．たとえば，2017 年 2 月現在，タンパク質配列データベース UniProt の RDF データセットは 277 億トリプルに達し[1,2]， 化合物データベース PubChem の RDF データセットは, 999 億トリプルに達しており[3,4]，いずれも頻繁に更新されている．

このような状況のもと，巨大な RDF データをトリプルストアへ効率よくロードする方法が求められている．効率がよい手法の代表的な一つとして，RDF データのファイルを分割し，並列にロードする方法が挙げられる．しかしながら，データセット内にブランクノードが含まれる場合，ブランクノードにはグローバルな ID が対応しないため, 別ファイルに分割されてしまうと，ロード後に同じノードとみなされなくなってしまうという問題があった．

そこで，同じブランクノードは同じファイルに含むよう，かつ，できるだけ分割後のファイルの最大サイズが小さくなるように，ファイルを分割するツール Split4Blank を開発した．本稿では，ツールの理論的背景となる問題定義および解法について述べる．また，生命科学分野のブランクノードを含むデータセット，および，ランダムあるいはスケールフリーネットワークの生成による架空データセットを用い，計算機実験を行い，ツールの計算時間に関し議論を行う．

2. RDF データセット分割問題

RDF ファイル分割ツールを設計するにあたり，以下の要件を考慮した．必須要件として，同一ブランクノードが別のファイルに分かれないようにすることが挙げられる．また，並列処理の際の処理時間を考慮すると，分割後のファイルの中で最大のファイルの大きさを最小化したい．

これらの要件から，ブランクノードに留意した RDF データセット分割問題(RDF Split for Blank)は以下のように定義される．ここで，トリプルの集合 T に対し，ノード r が主語あるいは目的語に現れるトリプルを含むとき，r は T に含まれるという．

インスタンス:トリプルの集合 T，正整数 m.

解: m 個の互いに素な T の部分集合 $T_1, \ldots, T_m$ で，$\cup T_i = T$ かつ，T_i に含まれるどのブランクノードに対しても $T_j (i \neq j)$にふくまれないもの．

尺度: $\max_i |T_i|$を最小化.

ここで，RDF Split for Blank を解く手法について概要を述べる．提案手法は主に二つの処理 SPLIT と COMBINE からなる．SPLIT は入力されたトリプルの集合 T から，同一ブランクノードが二つ以上の部分集合に含まれないよう，かつ，できるだけ小さい集合になるような分割を計算する．もう一つの処理 COMBINE は SPLIT で分割された小さなトリプルの集合群を組み合わせて，$\max_i |T_i|$を最小化するような分割 $T_1, \ldots, T_m$ を計算する．

アルゴリズム 1 は，処理 SPLIT を述べたものである．SPLIT はまず，トリプルを主語目的語共にブランクノード，片側のみブランクノード，ブランクノードを含まない，の三種類に分ける．さらに，主語目的語共にブランクノードのトリプルの集合に関し，連結成分を計算することで，同一ブランクノードが二つ以上の部分集合に入ることを避ける．最後に，片側のみブランクノードのトリプルやブランクノードを持たないトリプルを可能な限り小さく分ける．

アルゴリズム 1: 処理 SPLIT

入力: トリプルの集合 T

出力: T の分割 $T_1,\ldots,T_k$ ただし, 同一ブランクノードが二つ以上の部分集合に含まれない

0: $T_{b1} := \Phi$, $T_{b2} := \Phi$, $T_0 := \Phi$.

1: for each (s,p,o)∈T,
  if s, o 共にブランクノード
    $T_{b2} = T_{b2} \cup \{(s,p,o)\}$.
  else if s, o 共にブランクノードではない
    $T_0 = T_0 \cup \{(s,p,o)\}$.
  else
    $T_{b1} = T_{b1} \cup \{(s,p,o)\}$.

2: グラフ G=(V, E)を次のように構築する:
  $V = \{s \mid (s, p, o) \in T_{b2}\} \cup \{o \mid (s, p, o) \in T_{b2}\}$
  $E = \{(s, o) \mid (s, p, o) \in T_{b2}\}$.
  グラフ G 上の連結成分 $V_1,\ldots,V_k$ を計算する.

3. for each i ($1 \leqq i \leqq k$),
  Ti := {(s,p,o)∈T_{b2}|s か o が V_i の元}

4. for each (s,p,o) ∈T_{b1} .
  if s か o が Vi に含まれる
    Ti = Ti∪ {(s,p,o)}.
  else
    T_{k+1} ={(s,p,o)}, V_{k+1}={s か o かブランクノードの方}, k = k+1.

5. for each (s,p,o) ∈T_0
  T_{k+1} ={(s,p,o)}, k=k+1.

6. $T_1,\ldots,T_k$ を出力する

処理 SPLIT に関し，以下の命題が成り立つ.

命題 1: Ti と Tj (i ≠ j) 共に含まれるブランクノード b は存在しない.

証明: 各ブランクノード b について，もし，T_{b2} に含まれるならば，ステップ 2 で T_{b2} のノード集合 V は連結成分に分割されているため, T_{b2} で b を含むトリプルはすべて同一成分に分けられ，同一集合に含まれている．さらに，T_{b1} で b を含むトリプルは，一つのブランクノード b しか含まないため，ステップ 4 で一つの集合に含まれる．したがって，任意のブランクノード b は同じ集合に含まれる.

命題 2: 任意の Ti に対し, もし, T_{i1} と T_{i2} (ただし T_{i1}、T_{i2} ともに空集合ではない)に分割すると，T_{i1} と T_{i2} の両方に含まれるブランクノードが少なくとも一つ存在する.

証明: ステップ 5 で作られた Ti については，単集合のため，分割の片方が必ず空集合になり明らか．ステップ 4 で新たに作られた Ti については全てのトリプルに同一ブランクノードが含まれるため，どのように分割しても，分割後の空でない集合には同一ブランクノードが含まれる.

T_i がステップ 3 で連結成分 V_i より作られた場合，どちらも空集合でない T_i の分割 T_{i1} と T_{i2} で，T_{i1} と T_{i2} の両方に含まれるブランクノードが存在しないものがあると仮定する．ここで，$T_{i1}' = T_{i1} \cap T_{b2}$, $T_{i2}' = T_{i2} \cap T_{b2}$ とする.

T_{i1}' あるいは T_{i2}'が空の場合，T_{i1} あるいは T_{i2} は T_{b2} の部分集合である．ここで，T_{i1} を T_{b2} の部分集合とし，T_{i1} に含まれるブランクノードを b とする．このとき，T_i がステップ 3 で作られることより，T_{b2} は b を含み，命題 1 より，T_i は b を含むトリプルを全てもつため，T_{i2} は T_{b2} に含まれる b を含むトリプルを全て含む．これは仮定に矛盾する.

T_{i1}'，T_{i2}'のいずれも空でない場合，T_{i1} と T_{i2} の両方に含まれるブランクノードが存在しない仮定より，T_{i1}'と T_{i2}'の両方に含まれるブランクノードが存在しない．ここで，T_{i1}' に含まれるブランクノード b_1, T_{i1}' に含まれるブランクノード b_2 を考える．仮定より $b_1 \neq b_2$．一方，$T_{i1}' \cup T_{i2}'$ はブランクノードのみからなる連結グラフをなすため，$T_{i1}' \cup T_{i2}'$ において b_1 から b_2 への経路が存在する．経路のどこで分割しても，分割点はブランクノードであり，T_{i1}'，T_{i2}'の双方に含まれるため，仮定に矛盾する.

次に処理 COMBINE について述べる．COMBINE は小さく分けられた集合に対し，最大の集合を最小化するよう，集合を結合していく処理である．本処

理は，Multiprocessor scheduling 問題への近似アルゴリズムに基づいている．Multiprocessor scheduling 問題とは，ジョブの集合 T, プロセッサの数 m，ジョブの計算時間 length: T→Z+ に対し，終了時間 $\max_i \Sigma_{t\in T}$ length(t)を最小化する問題である．ジョブの集合をトリプルの集合族，プロセッサの数を分割数，ジョブの計算時間をトリプル数と対応させることにより，COMBINE 処理を Multiprocessor scheduling 問題に帰着させることができる．

Multiprocessor scheduling 問題は NP 困難であることが知られており[5]，数多くの性能が良い近似アルゴリズムが提案されているが，ツールにおいては計算時間を考慮して，[6]で提案された手法(Longest Processing Time 法)を採用した．この手法は単純な手法だが, 常に最適解の 4/3-1/(3m)倍以下の解を出力することが示されている．

アルゴリズム 2: 処理 COMBINE

入力:互いに素なトリプルの集合族 $T_1,\ldots,T_k$

出力: $S_1,\ldots,S_m$ ただし, 各 T_i に対し, S_j が存在して, $T_i \subseteq S_j$

0: for each i ($1 \leqq i \leqq m$) $S_i := \Phi$.

1: $T_1,\ldots,T_k$ をトリプル数の降順に並べ，$T_1',\ldots,T_k'$ とする．

2: j =1 から k まで
$|S_i|$が最小の i を選び，$S_i := S_i \cup T_j$

3. $S_1,\ldots,S_m$ を出力する

処理 SPLIT は各ステップ O(|T|)の計算時間を必要とするため，全体として O(|T|)時間である．処理 COMBINE はソートに O(|T|log|T|)時間を要するが，ツール上では，SPLIT の計算における T_0 を別にすることで，実用的にはかなり短くすることが可能である．

3. 計算機実験

3.1 実験内容

Split4Blank の実際上の計算時間の傾向を調べるため，生命科学分野でブランクノードを多く含む実際のデータセット，および，ランダムグラフ，スケールフリーグラフから生成した架空データセットで実験を行った．

生命科学分野のデータセットとしては，生命科学の略語辞書 Allie[7] (トリプル数 143,435,311), 化合物データ日化辞 RDF[8]の一部 (トリプル数 90,445,172) を利用した．これらのデータセットに対し，分割数 m を 2 から 10 まで変えて計算し, 計算時間を測定した．

架空データに関しては，三種類のグラフ生成方法を用いてグラフを発生させ，その頂点や辺に架空の URI およびブランクノードを貼り付けることで生成した．グラフ生成方法としては Erdős–Rényi G(n,p) ランダムグラフモデル[9], Watts-Strogatz モデル[10], Barabási–Albert モデル[11]の三モデルを利用した．Erdős–Rényi G(n,p)ランダムグラフモデルでは辺発生確率 p を 0.0005 に設定し, Watts-Strogatz モデルでは初期次数を 2，辺張り替え確率を 0.5 に設定し，Barabási–Albert モデルでは，初期頂点数を 2 とし，加辺パラメタを 30 に設定した. これらの設定の上で, 頂点数を 10,000 から 100,000 まで 10,000 ごとに変えてグラフを発生した．さらに各グラフへ URI やブランクノードによるラベル付けを行った．頂点へのブランクノードの発生割合は 0.5 とした．その後，各ラベル付きグラフからトリプルの集合を生成し，架空データセットを作成した．架空データセットについては，分割数を 2 と固定して，発生させた頂点数 10,000 から 100,000 までのグラフそれぞれについて計算時間を測定した．

3.2 実験結果

図 1, 2 は Allie および日化辞 RDF について，分割数 2 から 10 まで，計算時間を測定したものである．一つの分割数に対し，12 回試行を行い，最大値と最小値を除いた平均を示しており，縦軸が計算時間(ms)，横軸が分割数となっている．この二つの結果から，分割数に関しては，計算時間に影響が少ないことを見ることができる．

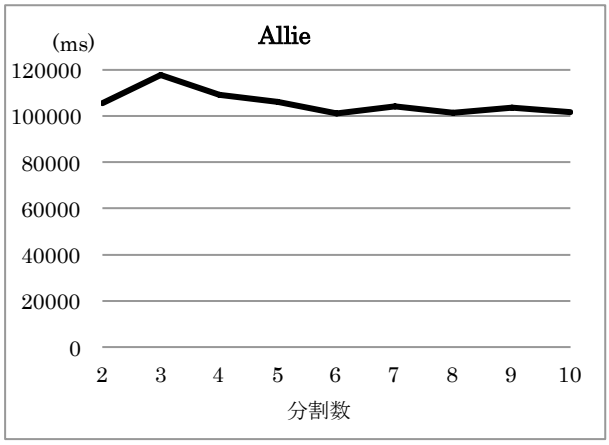

図 1: データセット Allie の分割計算時間

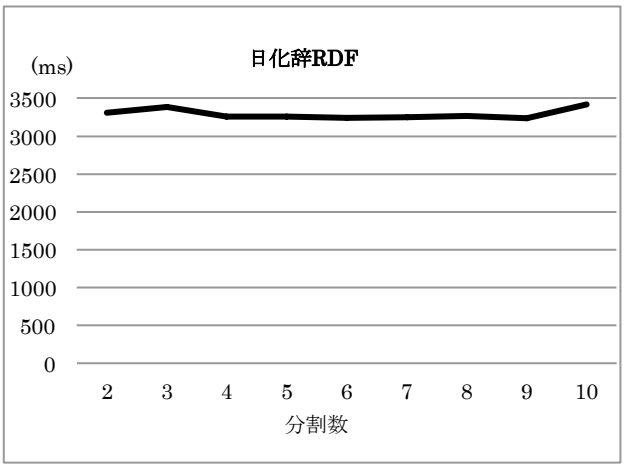

図 2: データセット日化辞 RDF の分割計算時間

図 3, 4, 5 は Erdős–Rényi G(n,p)ランダムグラフモデル，Watts-Strogatz モデル，Barabási–Albert モデルの結果である．いずれのモデルにおいても，頂点数が増えると計算時間は長くなっているが，急激な増え方ではなく，頂点数とほぼ比例する増え方になっている． これらの結果より，理論的な計算量の面のみならず，架空データセットを用いた計算機実験による計算時間の測定により，グラフの大規模化に対して，様々な構造のグラフにおいて，本手法は対応可能であると思われる．

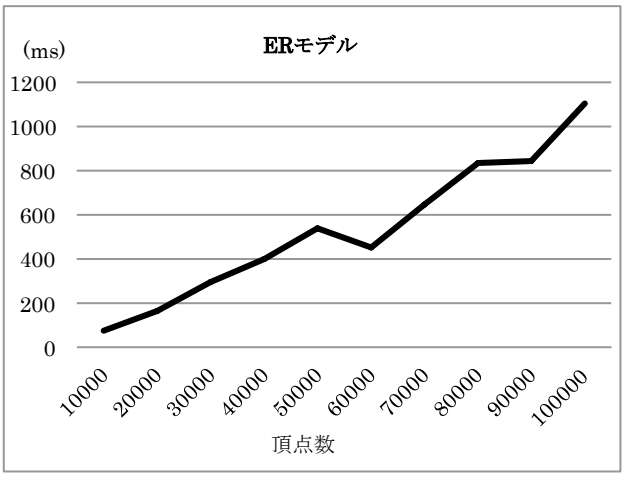

図 3: Erdős–Rényi G(n,p)ランダムグラフモデルから生成したデータセットの分割計算時間

4. まとめと今後の課題

ブランクノードを含む RDF データのファイルを分割し，並列にロードするためのツールとしてファイル分割ツール Split4Blank を開発した．その際，解くべき問題をトリプル数最大集合の最小化問題として定式化し，連結成分計算と Multiprocessor scheduling 問題の近似アルゴリズムを組み合わせることで効率よく計算するアルゴリズムを提案した．また，生命科学分野の実際のデータセットと，ランダムグラフやスケールフリーグラフの生成モデルを利用して生成した架空データセットを利用して，計算機実験を行い，計算時間について考察を行った．

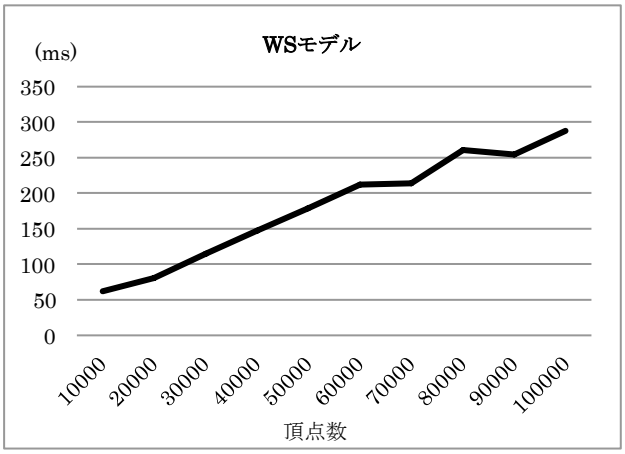

図 4: Watts-Strogatz モデルから生成したデータセットの分割計算時間

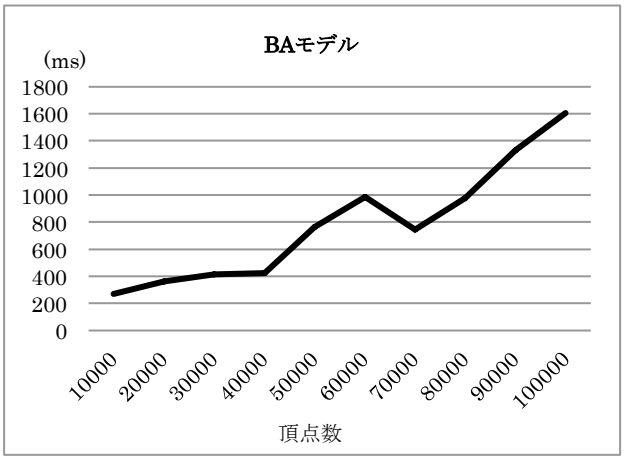

図 5: Barabási–Albert モデルから生成したデータセットの分割計算時間

今後の課題としては，より大規模化へ対応するために，ツールの並列化を検討したい．また，今回，架空データセットでは生成モデルの各パラメタを固定して行ったが，実データを調査し，より実データに近いパラメタで実験を行いたい．また，より多くの実データを試すことで，ツールの改善を図っていきたい．

参考文献：

[1] UniProt, http://www.uniprot.org/

[2] UniProt Linked Data, The EBI RDF Platform, https://www.ebi.ac.uk/rdf/services/uniprot/

[3] PubChemRDF, https://pubchem.ncbi.nlm.nih.gov/rdf/

[4] Fu, G., Batchelor, C., Dumontier, M., Hastings, J., Willighagen E., and Bolton, E.: PubChemRDF: towards the semantic annotation of PubChem compound and substance databases. Journal of Cheminformatics, 7(34), doi:10.1186/s13321-015-0084-4 (2015)

[5] Garey, M. R. and Johnson, D. S.: Computer and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, New York, 1979.

[6] Graham, R. L.: Bounds on multiprocessing timing anomalies. SIAM Journal on Applied Mathematics, 17 (2), 416–429 (1969)

[7] Yamamoto, Y., Yamaguchi, A., Bono, H., and Takagi, T.: Allie: a database and a search service of abbreviations and long forms. Database, doi: 10.1093/database/bar013 (2011)

[8] 木村考宏，櫛田達矢: 日化辞 RDF データの公開と化合物情報の統合. 情報管理 58(3), 204-212 (2015)

[9] Janson, S., Łuczak,T., and Rucinski, A: Random Graphs. John Wiley & Sons, Inc, New York, 2000.

[10] Watts, D. J. and Strogatz, S. H.: Collective dynamics of 'small-world' networks. Nature, 393(6684), 440-442 (1998)

[11] Barabási , A. L. and Albert, R.: Emergence of Scaling in Random Networks. Science, 286(5439), 509-512 (1999)