大規模グラフ計算用言語Fregelの二部グラフマッチングへの応用と拡張

江本研究室

栗木駿一

背景

近年大規模なグラフの解析の需要が高まっている
（例）SNSの解析
　　検索エンジンのページランク

グラフ解析のためのグラフ並列処理フレームワークが注目
（例）Fregel，Giraphなど

Fregel [Emoto+, ICFP 2016]

Pregel[Google 2010]の問題点を解決しようと提案された
関数型領域特化言語

- 頂点主体のグラフ並列計算モデル
- ユーザーが各頂点で繰り返される単一の計算を記述
- 頂点単位で並列にグラフ計算を行うことが可能
- 複雑なアルゴリズムでも簡単に記述可能

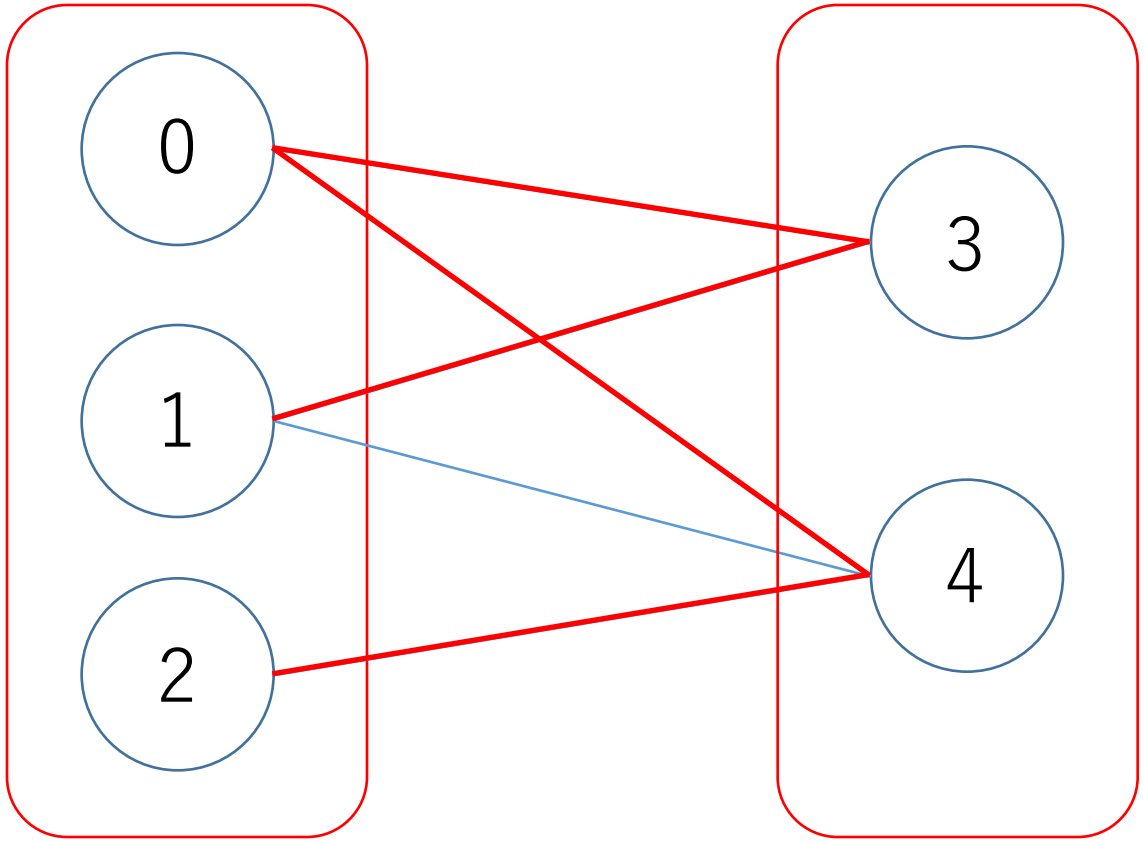
研究の目的と内容

・Fregelにおけるグラフ計算の記述例が少ない
→記述例を増やす

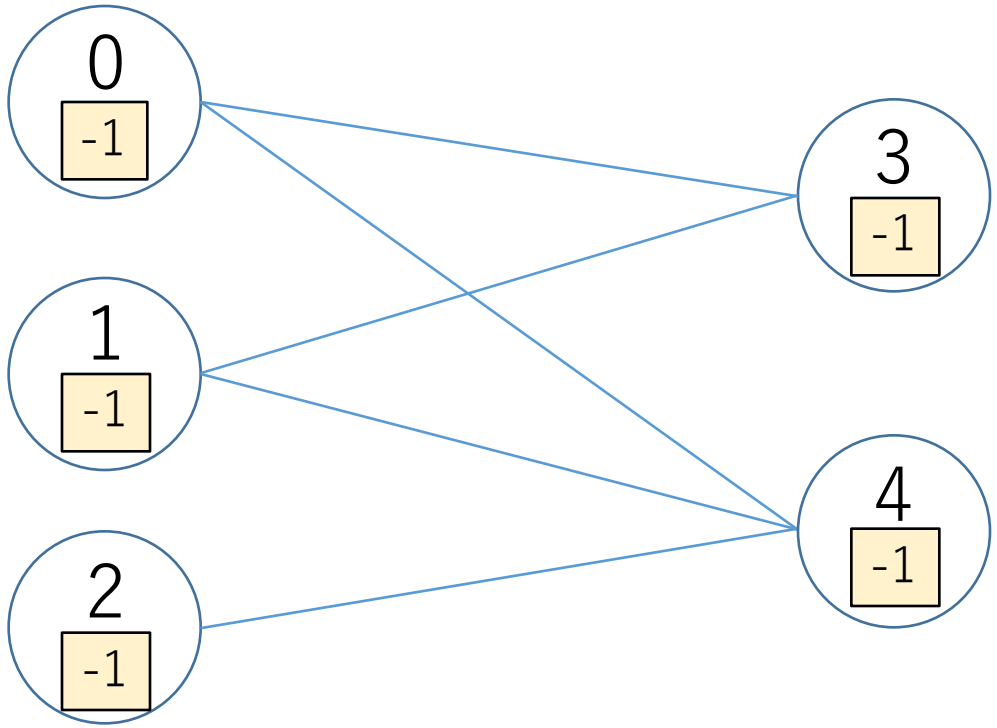
・Fregelで二部グラフマッチングを記述
→機能の拡張も行う

・拡張機能を用いたプログラムを用いて実験
→有用性，問題点を確認

二部グラフのマッチング

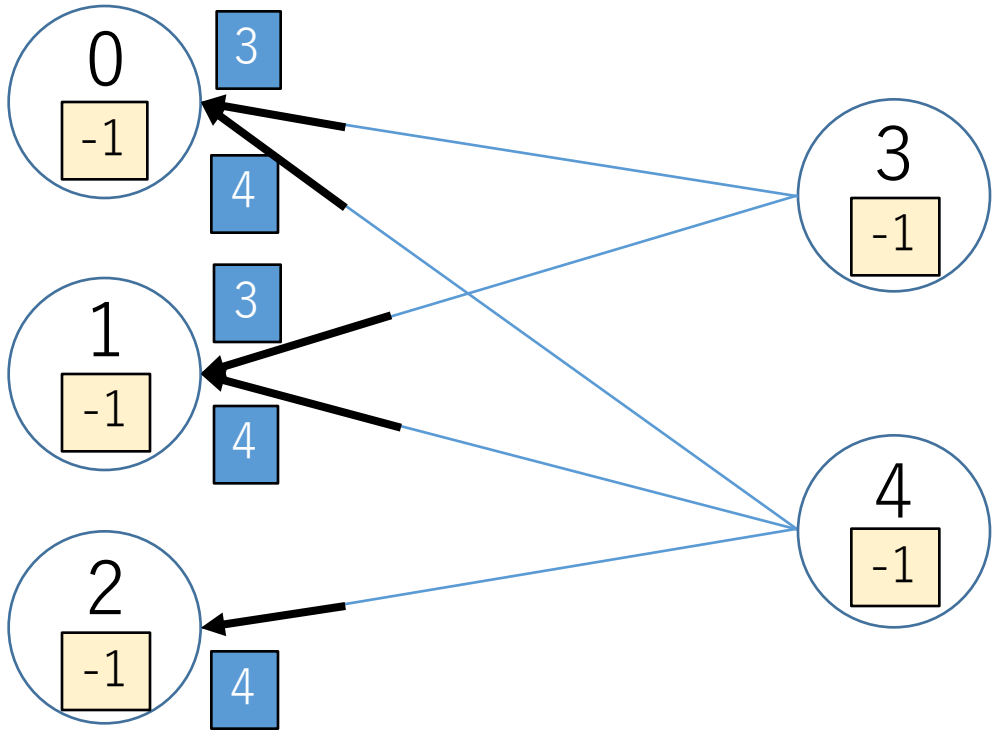


二部グラフマッチングのFregelアルゴリズム



STEP 1

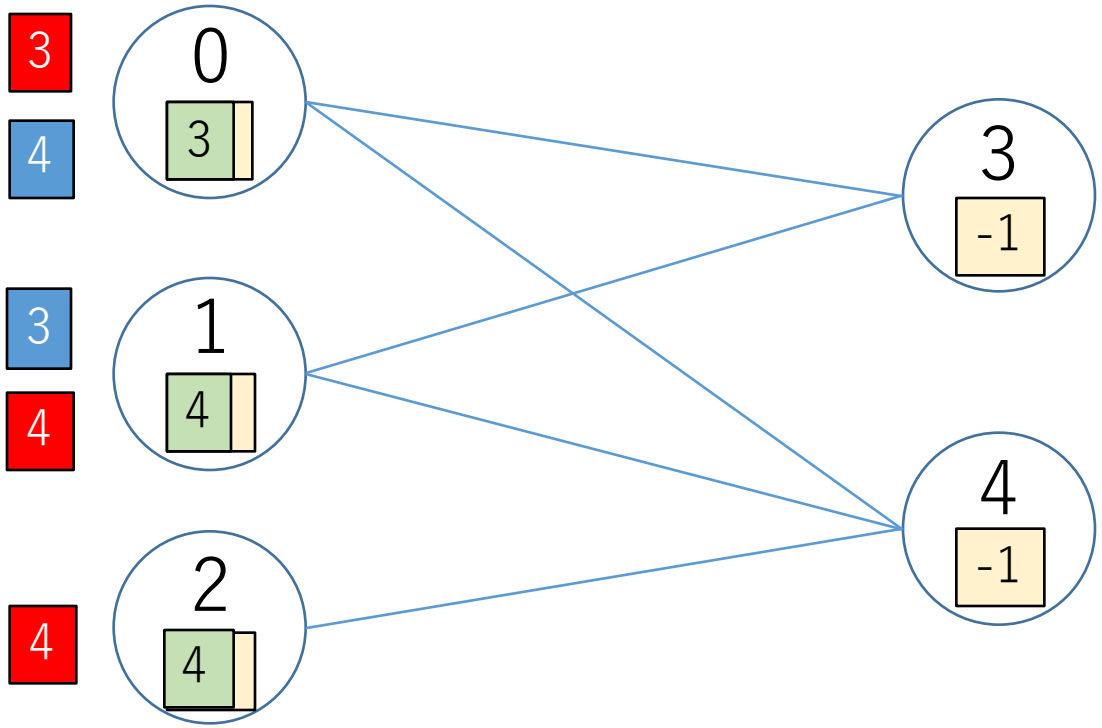
二部グラフマッチングのFregelアルゴリズム



STEP 1

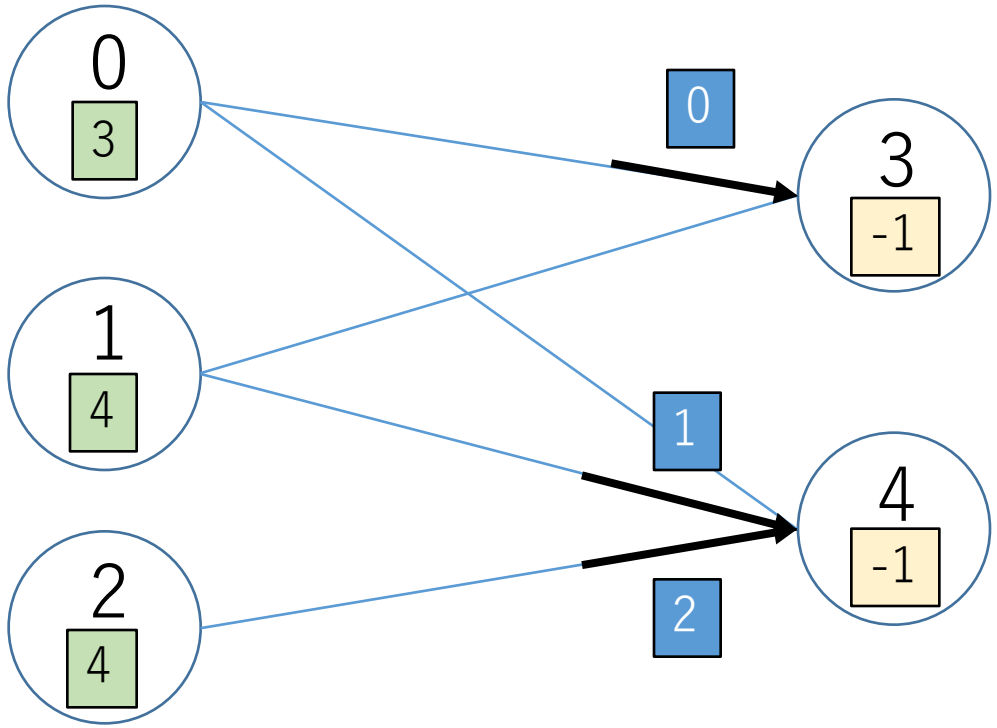
二部グラフマッチングのFregelアルゴリズム

ランダムに１つ選択



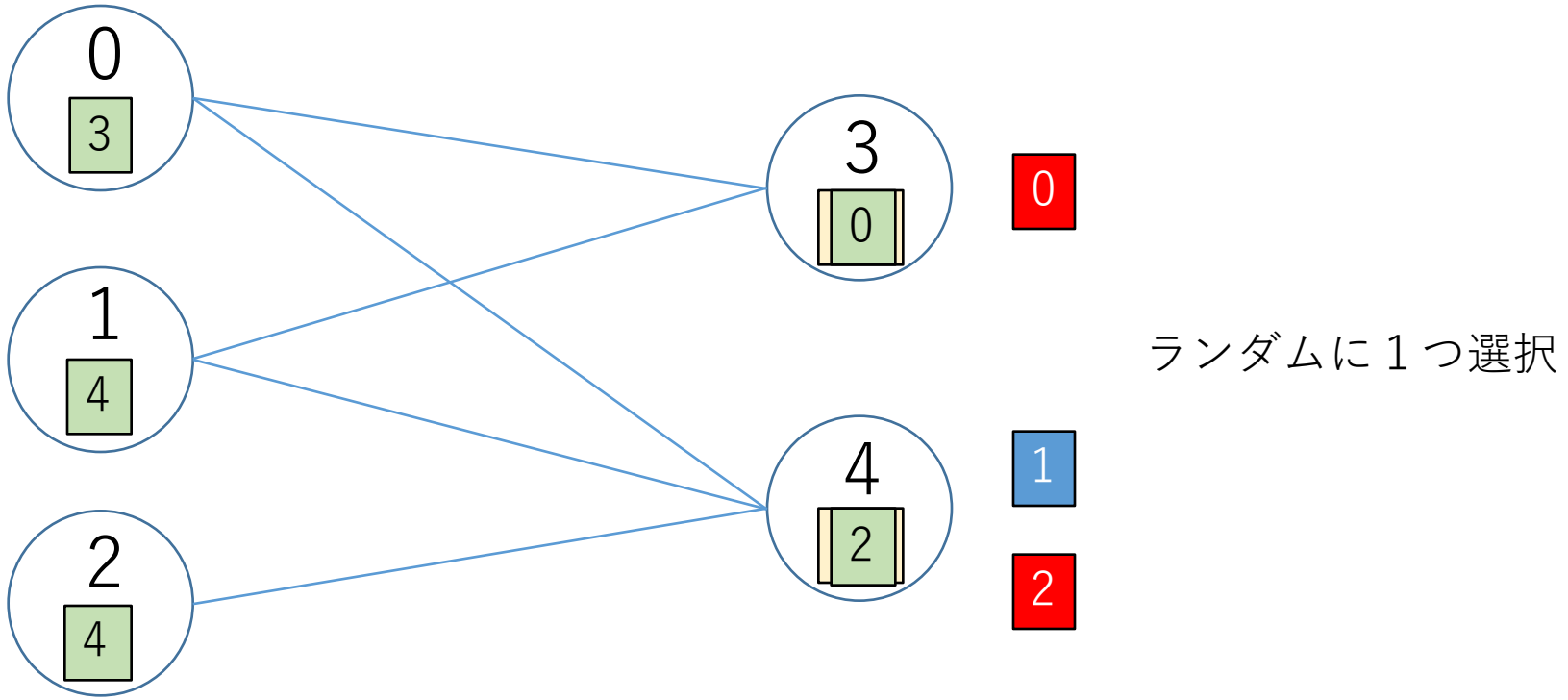
STEP 1

二部グラフマッチングのFregelアルゴリズム



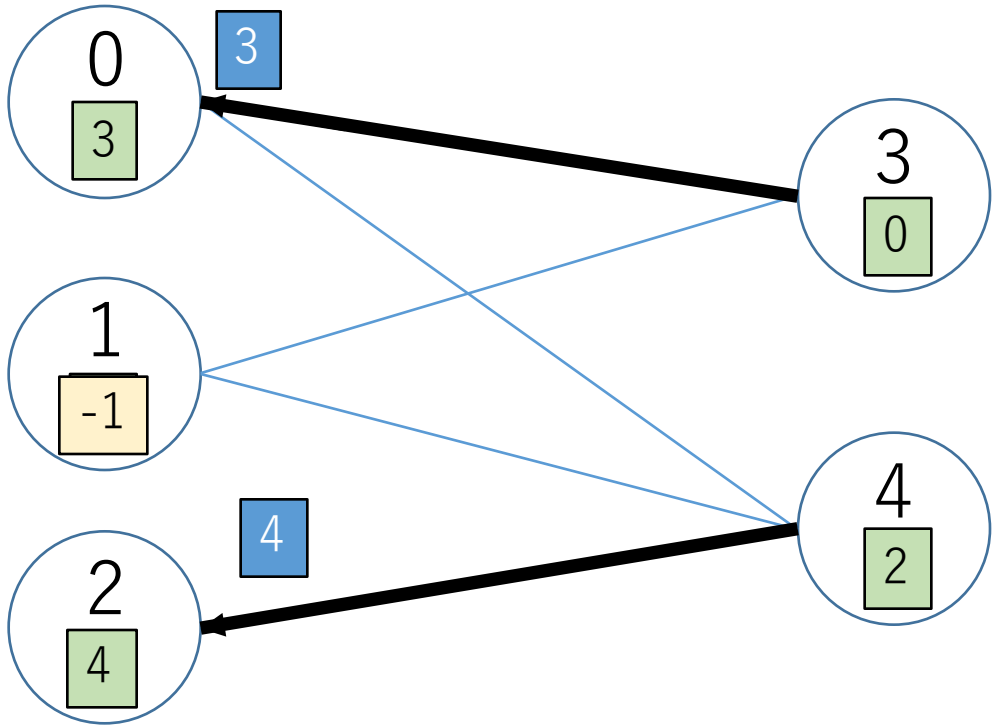
STEP 2

二部グラフマッチングのFregelアルゴリズム



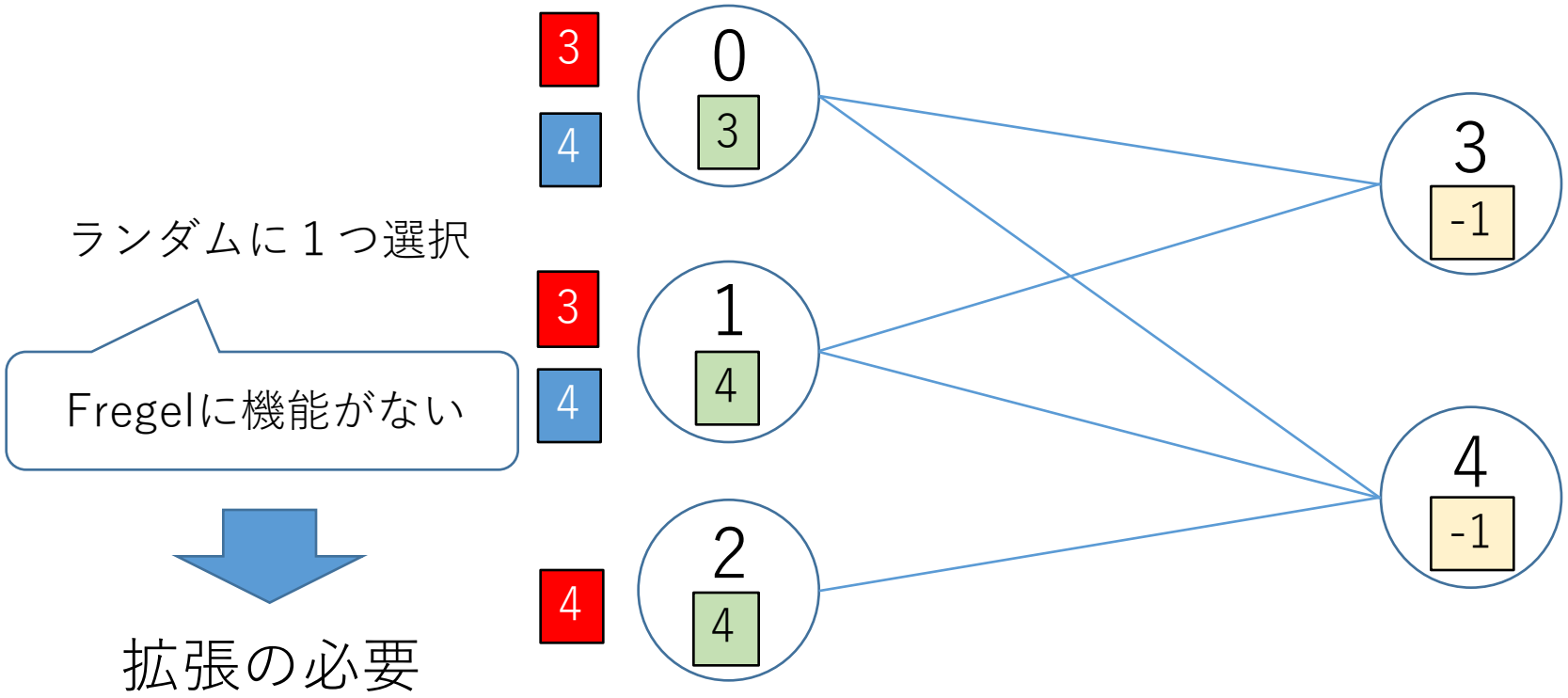
STEP 2

二部グラフマッチングのFregelアルゴリズム



STEP 3

二部グラフマッチングのFregelアルゴリズム



STEP 1

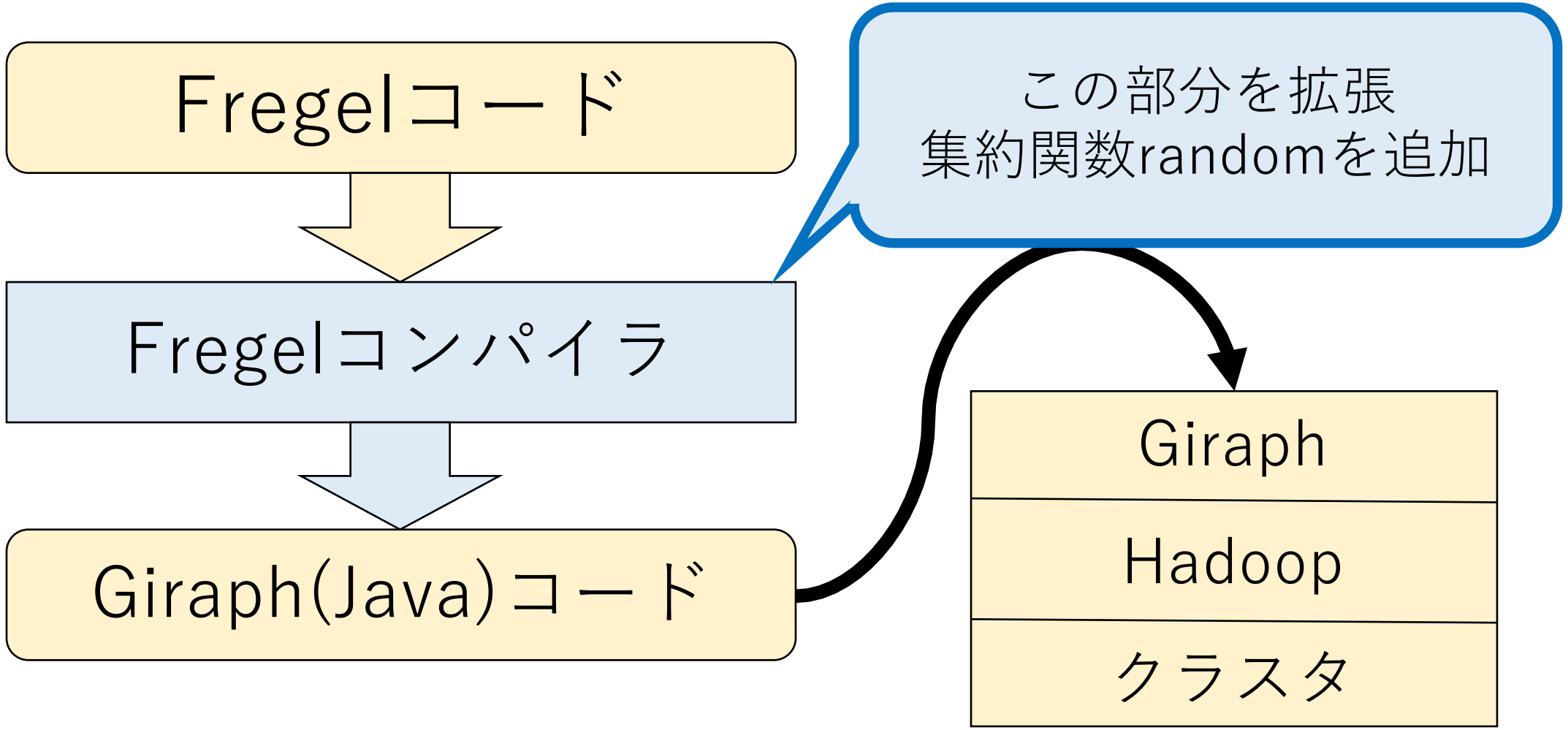
Fregelの二部グラフマッチング

```
bipartitematching g =
 let init v = -1;
    ss0 v = if (vid v > size/2  && val v.^set == -1)
            then random (-1) [vid u|(e,u)<-is v, val u.^set == -1]
            else val v.^match;
    ss1 v = if (vid v <= size/2 && val v.^set == -1)
            then random (-1) [vid u|(e,u)<-is v, val u.^set == vid v]
            else val v.^match;
    ss2 v = if (vid v > size/2 && val v.^set /= -1)
            then random (-1) [vid u|(e,u)<-is v, val u.^set == vid v]
            else val v.^match;
    step g = let g1 = gmap ss0 g;
                 g2 = gmap ss1 g1;
                 g3 = gmap ss2 g2
             in g3
 in giter init step Fix g
```

各ステップの処理

繰り返し実行

機能の拡張部分


Fregelコード
Fregelコンパイラ
Giraph(Java)コード
この部分を拡張
集約関数randomを追加
Giraph
Hadoop
クラスタ

実験内容

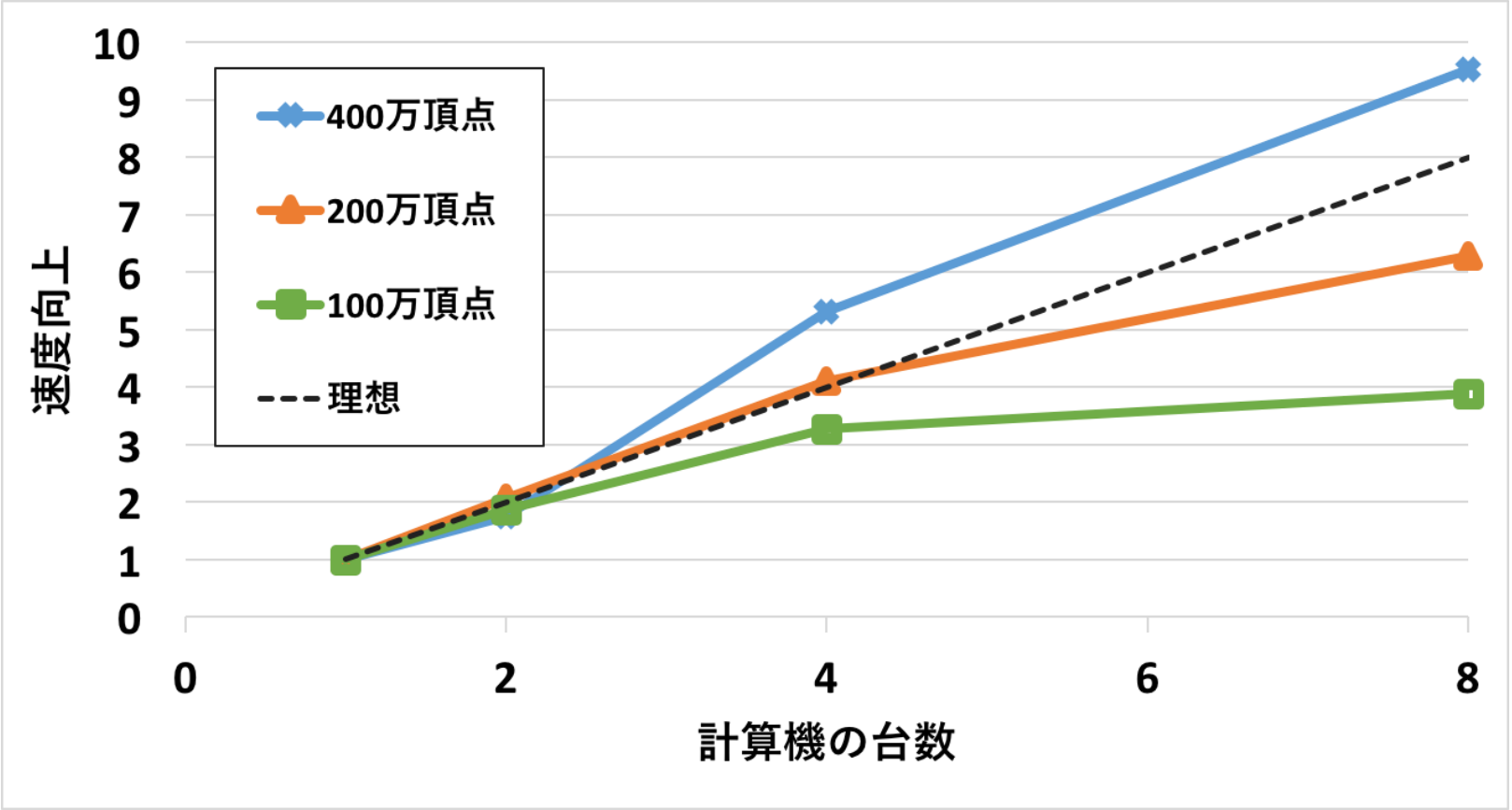
- 頂点数と計算機の台数を変化させて実行時間を測定
  計算機1台の実行速度をベースとした速度向上の様子を調べる

- 各頂点が10本のランダムな辺を持った二部グラフを使用

OS	Ubuntu 14.04.5 LTS
CPU	Intel(R) Core(TM) i5-6500 CPU @ 3.20GHz
メモリ	16GB (8GB×2, PC4-17000)
Java	Oracle JDK 1.8.0 131
Hadoop	Version 1.2.1
Giraph	Version 1.1.0

実験環境

実験結果



台数が増えるにつれて速度向上が改善している
→十分な並列性能が確認できた

まとめ

- 二部グラフマッチングをFregelで記述
- 記述に必要なFregelの機能の拡張
- 有用性の検証
  大規模なグラフほど台数増加による速度向上が顕著に
  →十分な並列性能を確認できた

- 今後の課題
  ・さらなる機能の充実
  ・Fregelの実行速度の向上

実行時間

計算機の台数	1	2	4	8
100万頂点	100.1	53.3	30.6	25.8
200万頂点	201.2	97.5	49.0	32.0
400万頂点	490.2	280.4	92.2	51.4

Fregelにおける二部グラフマッチングの実行時間(s)

計算機の台数	1	2	4	8
100万頂点	29.0	20.5	15.8	15.4
200万頂点	50.0	29.2	20.8	17.0
400万頂点	91.0	49.8	31.5	22.5

Pregelにおける二部グラフマッチングの実行時間(s)

拡張部分

```
static class Choicer{
    int n;
    Choicer(){ n = 0; }
    int choice(int x, int y) {
      n++;
      return Math.random() <= 1.0/n ? y : x;
    }}
```

オペレーターの準備

```
Choicer choicer = new Choicer();
  for (msg : messages)
    agg = choicer.choice(agg, (msg.agg));
```

使用箇所

- 整数nはyの数をカウント
- 1/nの確率でyを選択
- 初めは1/1，以降は1/2，1/3, …となる
  →最終的にどの要素も同じ確率で選ばれる